



COMPUTERIZED STRUCTURAL TESTING

Richard H. Dalrymple
Air Vehicle and Crew Systems Technology Department (Code 6043)
NAVAL AIR WARFARE CENTER
AIRCRAFT DIVISION WARMINSTER
P.O. Box 5152
Warminster, PA 18974-0591

13 December 1993

FINAL REPORT

Approved for Public Release; Distribution is Unlimited

Prepared for
NAVAL AIR SYSTEMS COMMAND
Department of the Navy
Washington, DC 20361-0001

1970 QUALITY INSPECTED 8

19970604 090

NOTICES

REPORT NUMBERING SYSTEM — The numbering of technical project reports issued by the Naval Air Warfare Center, Aircraft Division, Warminster is arranged for specific identification purposes. Each number consists of the Center acronym, the calendar year in which the number was assigned, the sequence number of the report within the specific calendar year, and the official 2-digit correspondence code of the Functional Department responsible for the report. For example: Report No. NAWCADWAR-92001-60 indicates the first Center report for the year 1992 and prepared by the Air Vehicle and Crew Systems Technology Department. The numerical codes are as follows:

CODE	OFFICE OR DEPARTMENT
00	Commanding Officer, NAWCADWAR
01	Technical Director, NAWCADWAR
05	Computer Department
10	AntiSubmarine Warfare Systems Department
20	Tactical Air Systems Department
30	Warfare Systems Analysis Department
50	Mission Avionics Technology Department
60	Air Vehicle & Crew Systems Technology Department
70	Systems & Software Technology Department
80	Engineering Support Group
90	Test & Evaluation Group

PRODUCT ENDORSEMENT — The discussion or instructions concerning commercial products herein do not constitute an endorsement by the Government nor do they convey or imply the license or right to use such products.

Reviewed By: _____


Branch Head

Date: _____

2/25/94

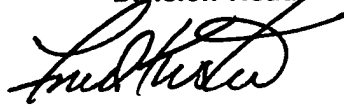
Reviewed By: _____


Division Head

Date: _____

2-28-94

Reviewed By: _____



Director/Deputy Director

Date: _____

3/1/94

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 13 Dec. 1993		3. REPORT TYPE AND DATES COVERED Final 1/87 - 9/92
4. TITLE AND SUBTITLE COMPUTERIZED STRUCTURAL TESTING			5. FUNDING NUMBERS	
6. AUTHOR(S) Richard H. Dalrymple				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Vehicle and Crew Systems Technology Department (Code 6043) NAVAL AIR WARFARE CENTER AIRCRAFT DIVISION WARMINSTER P.O. Box 5152 Warminster, PA 18974-0591			8. PERFORMING ORGANIZATION REPORT NUMBER NAWCADWAR-93085-60	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) NAVAL AIR SYSTEMS COMMAND Department of the Navy Washington, DC 20361-0001			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release; Distribution is Unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>The Experimental Research Section of the Aero Structures Division, Aircraft and Crew Systems Technology Department Naval Air Warfare Center, Aircraft Division, Warminster, PA is responsible for structural testing of a variety of different types and sizes of specimens. The purpose of these tests range from characterization of material systems to qualifying substructures for flight certification. Previously, simple tests were done with limited spectrum fatigue capabilities and little or no real-time data acquisition capability. More complicated tests were done on the lab's Hewlett-Packard HP-1000 computer. To expand the capability of the laboratory, five Hewlett-Packard HP-9826 desktop computers with HP-6940B multiprogrammers were purchased. These small desktop systems have all the capabilities of the HP-1000 but with a limited data acquisition and storage. Now a variety of tests can be performed on the desktop computers, from small static tests to spectrum fatigue tests, leaving the HP-1000 available for tests that require large data acquisition.</p> <p>It is not possible to generate generic software for use on a variety of test programs since the testing pedometers (loading configuration, data acquisition, fatigue spectrum, etc.) can vary significantly. Instead, software was developed specifically for one program, with the intent to identify test-specific requirements that can be modified for other programs.</p>				
14. SUBJECT TERMS Structural Test, Computer Program			15. NUMBER OF PAGES	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR	

ABSTRACT (Continued)

This paper documents software developed for the HP-9826 system to perform static and fatigue test for the High Strain Wing Repair Verification Program. For static tests, the software will control the load and monitor load feedback, and will initiate strain and extensometer data acquisition on the HP-1000 computer system. For random spectrum fatigue tests, the software will control and monitor the load as dictated by the fatigue spectrum data. Hardware and software validation routines are also documented in this paper.

TABLE OF CONTENTS

Abstract	i
Table of Contents	ii
List of Tables	iv
List of Figures	iv
Preface	vi
1.0 SCOPE	1
1.1 Introduction	1
1.2 Purpose	1
1.3 Document Overview	2
1.4 Background	2
1.5 Applicable Documents	5
2.0 REQUIREMENTS	8
2.1 System Requirements	8
2.2 Software Requirements	9
2.2.1 Core Software Requirements	9
2.2.2 Static Test Software Requirements	11
2.2.3 Fatigue Test Software Requirements	12
3.0 DESIGN	14
3.1 Definitions	14
3.2 Overview	14
3.2.1 CSCI-1 Test Software Architecture	15
3.2.2 CSCI-1 System Modes and States	16
3.2.3 CSCI-2 Architecture	17
3.2.4 CSCI-2 System States and Modes	19
3.3 CSCI-1 Design Description	19
3.3.1 CSC-1.1 Test Executive	19
3.3.2 CSC-1.2 Variable Definitions	23
3.3.3 CSC-1.3 Pump Commands	23
3.3.4 CSC-1.4 Load Commands	24
3.3.5 CSC-1.5 Test Parameters	25
3.3.6 CSC-1.6 Test Conditions	26
3.4 CSCI-2 Design Description	29
3.4.1 CSC-2.1 Program Check Pump Commands	29
3.4.2 CSC-2.2 Program Check Load Commands	29
3.4.3 CSC-2.3 Program Check Test Parameter Routines	30
3.4.4 CSC-2.4 Program Check Switches	31
3.4.5 CSC-2.5 Program Check Counters	31
3.4.6 CSC-2.6 Program Check Display	33

109706604-055

TABLE OF CONTENTS (Continued)

4.0	SOFTWARE USER'S GUIDE	38
4.1	Overview	38
4.2	CSCI-1 Test Software Execution Procedures	39
4.2.1	Startup Procedures	39
4.2.2	Main Menu Procedures	40
4.2.3	Static Test Procedures	44
4.2.4	Fatigue Test Procedures	46
4.2.5	Restart	49
4.3	CSCI-2 Verification Software Execution Procedures	49
4.3.1	Check Pump Commands	49
4.3.2	Check Load Commands	50
4.3.3	Check Test Parameter Routines	51
4.3.4	Check Switches	52
4.3.5	Check Counters	53
4.3.6	Check Display	54
APPENDIX A - HARDWARE DESCRIPTION		A-1
A.1	Description of test setup	A-1
A.2	Description of test equipment	A-2
A.3	Description of existing fatigue lab system	A-3
A.4	Description of existing HP-1000 system	A-4
A.5	Description of HP-9826 system	A-6
A.6	Description of the Hybrid HP-9826/HP-1000 system	A-9
APPENDIX B - CSCI-1 HIGH STRAIN WING REPAIR TEST PROGRAM LISTING		B-1
B.1	CSC-1.1 Test Executive	B-2
B.2	CSC-1.2 Variable Definitions	B-18
B.3	CSC-1.3 Pump Commands	B-21
B.4	CSC-1.4 Load Commands	B-24
B.5	CSC-1.5 Test Parameters	B-28
B.6	CSC-1.6 Test Conditions	B-37
APPENDIX C - CSCI-2 PROGRAM LISTINGS FOR VERIFICATION PROGRAMS		C-1
C.1	CSC-2.1 Check Pump Commands	C-2
C.2	CSC-2.2 Check Load Commands	C-6
C.3	CSC-2.3 Check Test Parameter Routines	C-13
C.4	CSC-2.4 Check Switches	C-24
C.5	CSC-2.5 Check Counters	C-28
C.6	CSC-2.6 Check Display	C-35

LIST OF TABLES

3.1. Counter Panel Switch Definitions	34
---------------------------------------	----

LIST OF FIGURES

1.1. HP-9826 Computer With HP-6940B Multiprogrammer	7
3.1. CSCI-1 Test Program	35
3.2. CSCI-2 Test Program	36
3.3. CSCI-1 Program Architecture	36
3.3a. Static Test States	36
3.3b. Fatigue Test States	36
3.4. Load Point Information	37
4.1. Autostart Request for Disk Labeled DAL40	56
4.2. Fatigue Test Main Menu	56
4.3. Static Test Main Menu	57
4.4. Request for Static Test Configuration File Name to Read	57
4.5. Request for Fatigue Test Configuration File Name to Read	57
4.6. Example of Error While Reading Configuration File	58
4.7. Menu for Changing the System Parameters	58
4.8. Query to Operator for the Design Limit Load	58
4.9. Query to Operator for the Maximum Test Load	59
4.10. Query to Operator for the Full Scale Voltage	59
4.11. Menu for Changing the Test Parameters	59
4.12. Query to Operator for the Loading Rate	59
4.13. Query to Operator for the Load Tolerance	59
4.14. Request for Static Test Configuration File Name to Write	60
4.15. Request for Fatigue Test Configuration File Name to Write	60
4.16. Example of Error While Writing Configuration File	60
4.17. Fatigue Test Main Menu with Pump Running	61
4.18. Static Test Main Menu with Pump Running	62
4.19. Load Calibration Display	62
4.20. Query to Operator for the Calibration Delay	63
4.21. Program Ended with No Configuration Changes	63
4.22. Configuration Change Notice During a Static Test	63
4.23. Static Test Ended with Configuration Changes Not Saved	64
4.24. Configuration Change Notice During a Fatigue Test	64
4.25. Fatigue Test Ended with Configuration Changes Not Saved	65
4.26. Query to Operator for the Number of Passes to Run	65
4.27. Hold at Zero Load During Static Test	66
4.28. Query to Operator for the Next Load Point	66
4.29. Example of Invalid Request for Next Load Point	67
4.30. Ramp to Load Point During Static Test	67
4.31. Hold at Load Point During Static Test	68
4.32. Example of Overload During a Static Test	68
4.33. Example of External Dump During a Static Test	69
4.34. Fatigue Test Starting Position Verification	69
4.35. Example of Operator Entry of New Starting Position	69

LIST OF FIGURES (Continued)

4.36.	Verification of Operator Entries	70
4.37.	Note to Operator That Counters are Being Updated	70
4.38.	Note to Operator That File Pointer is Being Updated	70
4.39.	Final Verification Prior to Starting Test	71
4.40.	Example of Display During Fatigue Test	72
4.41.	Example of Operator Interrupt During Fatigue Test	72
4.42.	Ramp to Zero Load Initiated by Operator Request	73
4.43.	Hold at Zero Load During Fatigue Test	73
4.44.	Example of Overload During a Fatigue Test	74
4.45.	Example of External Dump During a Fatigue Test	75
4.46.	Test Stopped Using Counter Panel Switches	75
4.47.	Pascal Operating System Command Line	75
4.48.	Operator Input to Execute the Check Pump Commands Program	76
4.49.	The Check Pump Commands Program Menu with Pump Off	76
4.50.	The Check Pump Commands Program Menu with Pump On	76
4.51.	Display After the Check Pump Commands Program Has Ended	77
4.52.	Operator Input to Execute the Check Load Commands Program	77
4.53.	Query to Operator for the Maximum Test Load	77
4.54.	Query to Operator for the Full Scale Voltage	77
4.55.	Query to Operator for the Output Load	78
4.56.	Example of Display During Ramp to Output Load	78
4.57.	Query to Operator for the Next Output Load While Holding	79
4.58.	Menu for Changing the Load Parameters	79
4.59.	Load Calibration Display	80
4.60.	Query to Operator for the Calibration Delay	80
4.61.	Display After the Check Load Commands Program Has Ended	81
4.62.	Operator Input to Execute the Check Test Parameter Program	81
4.63.	Check Test Parameter Program Fatigue Test Menu	82
4.64.	Check Test Parameter Program Static Test Menu	83
4.65.	Check Test Parameter Program Response to Menu Item '5'	84
4.66.	Check Test Parameter Program Response to Menu Item '6'	84
4.67.	Operator Input to Execute the Check Switches Program	84
4.68.	Example of Display With All Switches and Pump Off	85
4.69.	Example of Display With All Switches On	85
4.70.	Example of Display With Switches and Pump Randomly On and Off	85
4.71.	Display After the Check Switches Program Has Ended	86
4.72.	Operator Input to Execute the Check Counters Program	86
4.73.	Check Counters Program Menu at Start of Program	87
4.74.	Query to Operator for the Number of Times to Increment a Counter	87
4.75.	Example of Operator Entry of New Starting Position	87
4.76.	Verification of Operator Entries	88
4.77.	Check Counters Program Menu after Incrementing Counters	88
4.78.	Display After the Check Counters Program Has Ended	89
4.79.	Operator Input to Execute the Check Display Program	89
4.80.	Query to Operator for the Test Mode	89
4.81.	Display After the Check Display Program Has Ended in the Static Mode	90
4.82.	Display After the Check Display Program Has Ended in the Fatigue Mode	90

PREFACE

This report documents software to perform the static and fatigue tests for the High Strain Wing Repair Verification Program. The software was developed under the technical cognizance of Mr. J. Minecci and Dr. J. Alper. The author gratefully acknowledges Mr. V. Catone and Mr. M. Corrigan for their assistance with the test hardware and software, Mr. R. Hay for his help with some of the figures, Dr. D. Barrett for his technical review of the report, and Ms. L. Dalrymple for her invaluable assistance in organizing and reviewing the report.

1.0 SCOPE

1.1 Introduction

The Experimental Research Section of the Advance Structures Technology Branch, Aero Structures Division at the Naval Air Warfare Center, Aircraft Division, Warminster, is the Navy field activity responsible for performing a variety of tests related to aircraft structures. Previously, simple tests were done with limited spectrum fatigue capabilities and little or no real-time data acquisition capability. More complicated tests were accomplished using the facility's Hewlett-Packard HP-1000 main-frame computer. To expand and improve the capability of the facility, five Hewlett-Packard HP-9826 desktop computers with HP-6940B multiprogrammers were purchased. These small desktop systems have all the capabilities of the HP-1000 but with limited data acquisition and storage. Now a variety of tests can be performed on the desktop computers, from small static tests to spectrum fatigue tests, leaving the HP-1000 available for tests that require large data acquisition. The software for the desktop computers was written in Pascal using the HP-Pascal operating system.

1.2 Purpose

The purpose of the system is to take advantage of HP-9826 desktop computers to off-load the mainframe computer for the test floor, and to add computer driven test with data retrieval capability to the fatigue lab.

1.3 Document Overview

This report discusses the software for the HP-9826 to control the testing and perform the data retrieval. While the design of the software for the High Strain Wing Repair Verification Program is described in this report, the requirements and architecture are generic to any test software. Military Standard DOD-STD-2167A, "Defense System Software Development," was used as a guide for the format of this report.

1.4 Background

The Experimental Research Section conducts tests for characterization of new materials and for the structural adequacy of aircraft subcomponents. The characterization of materials testing is typically performed in a laboratory, commonly referred to as the "fatigue lab". The majority of aircraft subcomponent tests are performed in the structural test area, or "test floor."

The fatigue lab is a 1000 square foot, environmentally controlled area. There are six servo-hydraulic test fixtures contained in the fatigue lab, with the associated electronic hardware to control the test fixtures. Most of the electronic hardware is dedicated to imparting the desired load on the material being tested, and reading the actual load from the test fixture. One hardware component, the function generator, outputs a variety of waveforms to cause the test fixture to run a fatigue test. For example, a simple ramp function can be used to perform a quasi-static test, or a continuous sine-wave can be used for a constant amplitude fatigue test. Another hardware component, the profiler, allows a limited series of loads to be stored on magnetic tape, then controls the test fixture to use the loads on the tape

in sequence. The sequence can be repeated as many times as desired. These hardware components are obviously limited to the tests described. In addition, these components do not have provisions for recording any material parameters, such as strain.

The test floor is a 20,000 square foot area inside a hanger and not environmentally controlled, i.e., it is heated in the winter, but not air-conditioned in the summer. The test floor is essentially an open area where specialized test fixtures can be fabricated. Numerous servo-hydraulic actuators, ranging from 5,000 to 100,000 pound load capacity, are available for building into the test fixture to apply loads to the aircraft subcomponent.

The control room, a 2000 square foot environmentally controlled area, contains the same type of electronic hardware to control the tests as does the fatigue lab, just more of it. The heart of the control room is a Hewlett-Packard HP-1000 computer which runs the actuator controllers. The HP-1000 has the versatility to run any type of desired test. It can perform a quasi-static or constant amplitude test like the function generator. The HP-1000 can also perform a random spectrum fatigue like the profiler, but the HP-1000 is not limited in the number of load levels that can be stored. Another benefit of the computer-based system is the capability to record test parameters at any time during a test. Loads, strains, deflections, temperatures, etc., can be recorded instantaneously, and used later for the post-test analysis.

The overriding disadvantage of using the HP-1000 as a controller on the test floor is that there is only one of them, and it is prohibitively expensive to purchase another one. When the computer is controlling one test, it cannot control another

test, it cannot be used for post-test analysis, and it cannot be used for the software development of a future test. This could be corrected by using a time-share operating system, but the computer system was purchased before time-share systems were available on a computer of this capacity. The other reason a time-share system is not used is because the computer must be dedicated to perform tests of the size that run on the test floor. Typically, the components being tested are one-of-a-kind test articles costing millions of dollars to design and fabricate. If a time-share system was used, the through-put of the test would be effected. The limiting factor for the test time is usually a combination of the computer through-put and the response of the hydraulic actuators. A time-share system also allows the possibility of a user to hang the system. If the system were to hang while a subcomponent were at a high load, the time required to catch and correct the problem could cause the subcomponent to fail.

The solution to the limited capabilities of the fatigue lab controllers, and the one-program-at-a-time HP-1000 test floor was to use the new generation of desktop computers. Desktop computers had the capability of through-put to perform the functions of the HP-1000, but for years did not have the associated I/O capabilities until Hewlett-Packard developed the HP-6940B multiprogrammer. Tied to an HP-9826, the multiprogrammer has the capability to perform the same functions of the HP-1000, but on a smaller scale. The new desktop system, as shown in Figure 1.1, has enough capability to control the less complex tests on the test floor, and has all the capability required to perform tests in the fatigue lab, at a cost that allows several systems to be purchased. Five computer/multiprogrammer systems were purchased by the Experimental Research Section to perform these functions.

For a detailed description of the test and computer hardware, refer to Appendix A.

1.5 Applicable Documents

The following manuals are published by the Hewlett-Packard Desktop Computer Division, Fort Collins, Colorado, except where noted:

- **The Pascal Handbook for the HP 9826 and 9836 Computers, HP Part Number 09826-90071, 1982.**
- **The Pascal User's Manual, HP Part Number 09615-90020, 1983.**
- **The Pascal Procedure Library User's Manual for the HP 9826 and 9836 Computers, HP Part Number 09826-90075, 1982.**
- **HP 9122D/S and 9123D/S Disc Drives Operator's Manual, HP Part Number 09122-90000, 1985.**
- **Multiprogrammer Model 6940B Operating and Service Manual, HP Part Number 06940-90008, 1979.**
- **Programming Guide, HP 9826-GPIO(Basic) with the 6940B Multiprogrammer, HP Part Number 06940-90010, 1983.**
- **D/A Voltage Converter Card Model 69321B Operating and Service Manual, HP Part Number 69321-90003, 1972.**
- **Quad D/A Voltage Converter Card Model 69322A Operating and Service Manual, HP Part Number 69322-90001, 1977.**

- Relay Output Card Model 69330A Operating and Service Manual, HP Part Number 69330-90001, 1970.
- High Speed A/D Converter Card Model 69422A Operating and Service Manual, HP Part Number 69422-90001, 1978.
- Low Level A/D Converter and Scanner Card Model 69423A Operating and Service Manual, HP Part Number 69423-90001, 1979.
- Event Sense Card Model 69434A Operating and Service Manual, HP Part Number 69434-90001, 1972.
- Minicounter II Model 428 Instruction Manual, Manual Number IM 428, Waugh Controls Corporation, Chatsworth, CA, 1977.

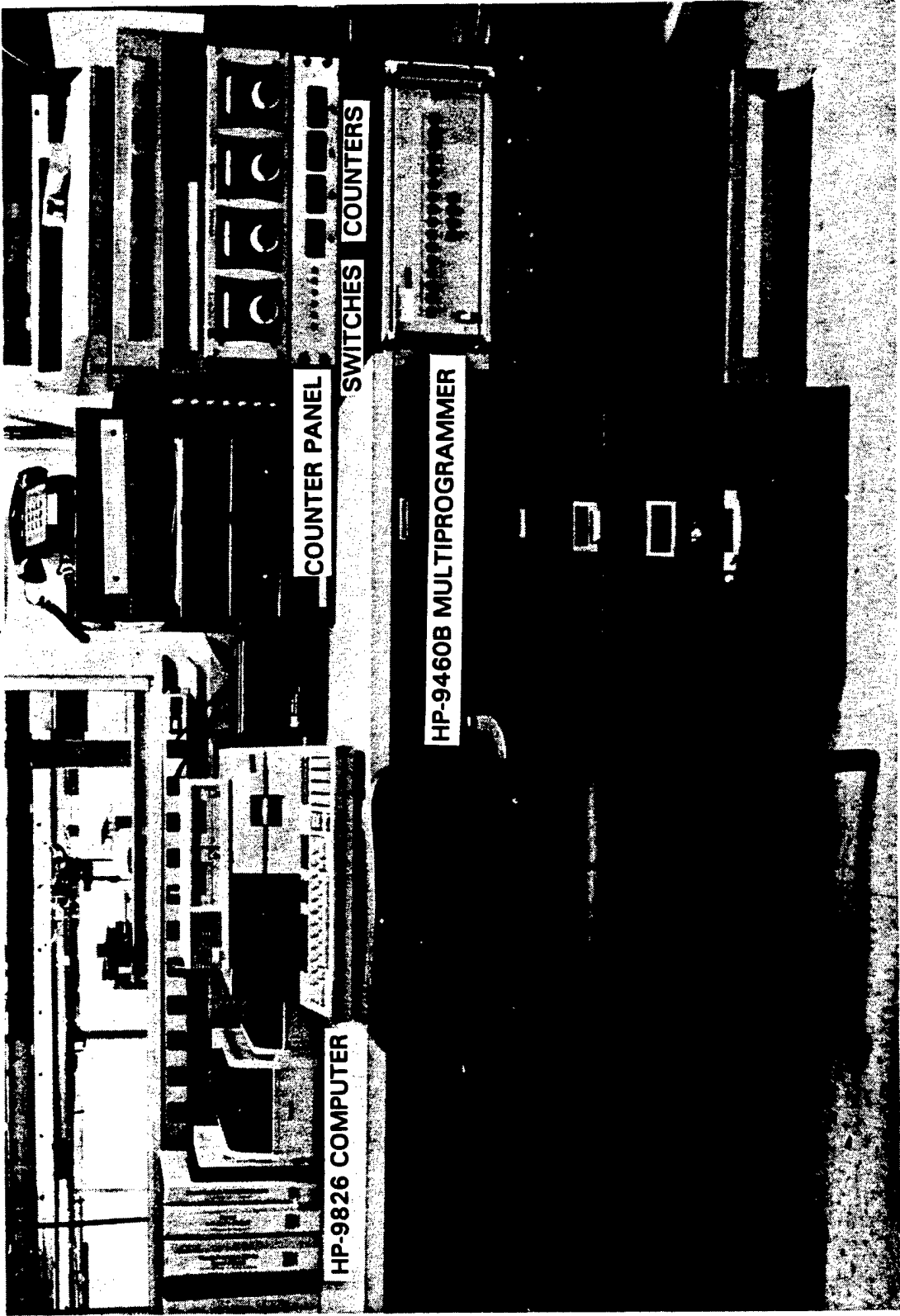


Figure 1.1. HP-9826 Computer With HP-6940B Multiprogrammer

2.0 REQUIREMENTS

The requirements are divided into system requirements and software requirements. The system is defined as all the hardware and software required to run a test. The hardware includes the HP-9826 computer, the HP-6940B multiprogrammer together with it's associated cards, and any other wiring and electronic devices (see Figure 1.1) required for the computer and multiprogrammer to interact with the test instrumentation (load controllers, strain gages, etc.). The software requirements are divided into core, static test, and fatigue test requirements. The core software is software that is common to both a static and fatigue test. Test specific requirements are given in the static and fatigue test requirement sections.

2.1 System Requirements

2.1.1 The system must be tied into the control panel for the hydraulic power supply, or pump, to allow the following capabilities:

2.1.1.1 The system must have the capability to allow the operator to turn on the pump, and to prevent the operator from turning on the pump. This permits the software to allow the pump to be turned on only after setting a prescribed state, such as zero load for the actuators.

2.1.1.2 The system must be able to dump the pump.

2.1.1.3 The system must sense when the pump is dumped externally.

2.1.2 The system must have the capability to read voltages, such as load feedback voltage.

2.1.3 The system must have the capability to output voltages, such as load command voltage. The system must also have the capability to remove the voltage slowly in the event that the computer software hangs up.

2.2 Software Requirements

2.2.1 Core Software Requirements

2.2.1.1 The software must be able to determine the state of the pump. If the pump is dumped externally during a test, the software must suspend the test at that point and return control to the operator.

2.2.1.2 The software must be able to control the state of the pump. At the start of the program, and after the configuration is set, the software must send a zero load output to the Servacs (the Servac is the electronic interface between the operator and the hydraulic actuator, and is described in Appendix A). The software must not allow the pump to be started until after this condition is set, and the load feedback from all actuators is within tolerances.

2.2.1.3 The software must be able to convert the desired load in pounds to volts. The conversion depends on the actuator size and load range of the Servac. The software must be compatible with the size of the actuators and the load ranges of the Servacs. For the test floor, the software must allow the operator to type in the capacity of the actuators. For the fatigue lab, the capacity must be selectable for 20, 50, or 100 kip at 10, 20, 50, or 100% load range.

2.2.1.4 The software must be able to read the feedback, or "B" bridge of the load cell, from any load cell on the test floor or fatigue lab.

2.2.1.5 The software must ensure that the feedback follows closely to the desired load. Any test where the feedback is lagging significantly behind the desired load can indicate a failure mode in the test element, or a failure in the test fixture. The software must allow the operator to change the amount of tolerance allowed between the desired load and the feedback. This is important in that, for every test, the amount of lagging between the desired load and the feedback can never be predicted, it is more of a trial and error procedure performed by the test engineer.

2.2.1.6 The software must allow the operator or test engineer to stop the test at any point. This is extremely important in a situation where a test anomaly has occurred. For example, if a noise comes out of the test element, the engineer may want to stop the test to check the strain data. Another reason to stop a test immediately is due to a hydraulic leak. The software must also give the operator the capability to immediately ramp the output load to zero, or resume the test from where it was stopped. If the operator chose to ramp the load to zero, the software must give the operator an option to dump the pump, as in the example of a hydraulic leak.

2.2.1.7 If data is recorded during the test, the software must have the capability to print the data.

2.2.1.8 In addition to the pump and load feedback tolerance checks mentioned above, all software shall be written with safety in mind. This is not only important to prevent an inadvertent failure of the test element, but also the safety of those individuals working in the area of the load frames or test fixtures; an unanticipated

catastrophic failure can cause great personal injury. Therefore, the software must be designed and written with safety as the most important factor.

2.2.2 Static Test Software Requirements

For any static test program, specialized software could be written to perform the test. However, in the fatigue lab, many static tests are very similar in nature, and therefore, a generic static test program can be written. Some options can be built into the software to give it the maximum application. The following are requirements that must be programmed into generic static test software:

2.2.2.1 The software must have a procedure for entering a maximum test load, or an option to run the test to failure, in which case the maximum test load would be the capacity of the hydraulic actuator.

2.2.2.2 The software must have the capability to read strain gages or extensimeters. A generic static test program need only read a maximum of five strain gages, any more than that would slow down the system considerably as described in Appendix A. The software must also allow for calibration of the strain gages. Calibration is performed by an electronic device that requires the operator to adjust the settings. The software must have the capability to show an operator the strain reading during calibration.

2.2.2.3 Even though the system is capable of reading load and strain data at each small increment of load, the software must have the capability to load in steps to the maximum load. This gives the engineer the option of reading data at percentages of the maximum load for easier data reduction after the test. In addition, the software must retain the most recent incremental loads and strains in

the event of a failure. For example, the test engineer may want load and strain data at every 1,000 pound increment. During the test, while loading from 5,000 to 6,000 pounds, the test specimen fails. The software should retain the last ten load and strain readings so that the engineer can see what occurred at the onset of failure.

2.2.2.4 The program must have the capability to run a tension or compression test in either load or displacement control. Some Servacs have the capability to control the hydraulic actuator based on displacement rather than load.

2.2.2.5 The software must allow the operator to choose a straight ramp or sine wave function for loading.

2.2.2.6 The software must allow the operator to print data during the test, so that the load and strain data can be interpreted during the test. In any case, the software must store the data for printing or plotting after the test.

2.2.3 Fatigue Test Software Requirements

In general, no two fatigue test programs are ever the same. First and foremost, no two fatigue test spectrums are ever the same. Even in a test using one spectrum, the test engineer may want to remove certain load levels in the spectrum to determine what effect that may have. Therefore, a generic random spectrum fatigue test program cannot be written. However, software can be written specifically for one application, but annotated where test program specifics are required to facilitate rewriting the software for another test program. Software requirements for a fatigue test are as follows:

2.2.3.1 The software must allow the use of a fatigue spectrum.

2.2.3.2 The software must allow the test to be started at any point in the spectrum, not just the beginning.

2.2.3.3 The software must be written to allow the test to be stopped at any point given by the test engineer. This is typically done for inspection intervals on the test element, or to stop the test for the night.

2.2.3.4 The software must be written to run completely unattended day or night, even over weekends, providing nothing unusual occurs.

3.0 DESIGN

This section discusses the test program software in top-level detail.

3.1 Definitions

3.1.1 Computer Software Configuration Item (CSCI): A configuration item for computer software. this project has two CSCIs, numbered CSCI-1 and CSCI-2.

3.1.2 Computer Software Component (CSC): A distinct part of a CSCI. CSCs may be further decomposed into other CSCs and CSUs. CSCs are numbered such that they can be traced to the parent CSCI. For example, the first two CSCs contained in CSCI-1 are numbered CSC-1.1 and CSC-1.2.

3.1.3 Computer Software Units (CSU): An element specified in the design of a CSC that is separately testable. CSUs are numbered such that they can be traced to the parent CSC. For example, the first two CSUs contained in CSC-1.2 are numbered CSU-1.2.1 and CSU-1.2.2.

3.2 Overview

There are two CSCIs for the High Strain Wing Repair Structural Verification Program. CSCI-1 is the test software to perform the static and fatigue tests (see Appendix B for a program listing). The software was developed by first writing generic test software, then tailoring certain aspects to the peculiarities of the High Strain Wing Repair tests. CSCI-2 is the verification software used to check both the hardware and software (see Appendix C for a program listing). This is accomplished by using portions of CSCI-1, then revising them to run as a stand

alone program to verify the function of the software in CSCI-1 and to verify the system hardware.

3.2.1 CSCI-1 Test Software Architecture

To simplify development of the test software, CSCI-1, the software is segregated into six CSCs, the test executive software and five library modules, as shown in Figure 3.1 and described in sections 3.2.1.1 through 3.2.1.5 below. The library modules contain all of the constants, functions, and procedures required to compile and run the test executive. These library modules are mostly unaltered from the generic test software, some having less alterations than the others. Once the library modules are written and debugged, they are compiled for use by the test executive. The intent is to alter the library modules only as required during software development. Having the library modules already compiled dramatically shortens the compile time of the test executive.

3.2.1.1 The Test Executive, CSC-1.1, executes the test by allowing the operator to select the static or fatigue test mode, reading in default values for the test parameters, allowing the operator to modify the test parameters as required, and then running the test.

3.2.1.2 The Variable Definitions library module, CSC-1.2, defines all of the constants used in the test executive and in the other library modules.

3.2.1.3 The Pump Commands library module, CSC-1.3, contains the procedures relating to the hydraulic power supply.

3.2.1.4 The Load Commands library module, CSC-1.4, contains the procedures relating to controlling and reading the load applied to the test specimen.

3.2.1.5 The Test Parameters library module, CSC-1.5, contains the procedures to allow the operator to read, change, and save the test parameters, and to restart the fatigue test at any point in the spectrum.

3.2.1.6 The Test Conditions library module, CSC-1.6, contains the procedures to check switches during the test in case the operator has thrown a switch, and all the procedures necessary to operate the electronic counters which have a battery backup in the event of a power loss.

3.2.2 CSCI-1 System Modes and States

3.2.2.1 CSCI-1 has two basic modes, the static test mode and the fatigue test mode. The program defaults to fatigue test mode when started. After that, the operator can change modes during the menu state, described below.

3.2.2.2 Both modes have the same three states: the menu state, the execution state, and the operator input state.

3.2.2.2.1 The menu state is the operators avenue to change the test mode, any of the test parameters, power up the hydraulic power supply, calibrate the load readings, start the test, or end the program.

3.2.2.2.2 During the execution state, the test is actually performed, i.e. applying load to the test specimen, reading the load feedback, checking that the load feedback is within the set parameters, checking that the hydraulic power supply is still running, etc. The execution state continues to run without operator input until

a specific event occurs. In the static test mode, the execution state will continue until the prescribed load level is reached, at which time the execution state is stopped, and the operator input state is entered. In the fatigue test mode, the execution state will continue until the prescribed point in the spectrum is reached, at which time the execution state is stopped, and the operator input state is entered. Lastly, during either mode, if the operator desires that the test be stopped, the operator can enter any key, which will halt execution and enter the operator input state.

3.2.2.2.3 During the operator input state, the load is held constant until the operator enters an appropriate response, identified below. The load is continually monitored to be certain the load feedback is within the set parameters. If the feedback does not stay within those parameters, the pump is shut down and the output load is quickly ramped to zero load, then the program returns to the menu state.

3.2.2.2.3.1 If the operator input state is entered while in the static test mode, the operator is requested to enter the next load value.

3.2.2.2.3.2 If the operator input state is entered while in the fatigue test mode, the operator is asked whether to continue or quickly ramp to zero load. If the operator request is to ramp to zero load, the option is given to shut down the hydraulic power supply.

3.2.3 CSCI-2 Architecture

To verify the operation of the hardware and software, CSCI-2 was developed using selected software from CSCI-1. CSCI-2 consists of six CSCs, as shown in Figure

3.2 and described in sections 3.2.3.1 through 3.2.3.5 below, each of which is a stand-alone program, so that only the subject tasks are tested. Not all aspects of CSCI-1 are tested; only those components that are easily separable and are prone to configuration changes have been developed into verification routines. Any errors or configuration changes required as a result of the CSU testing must be reflected in CSCI-1.

3.2.3.1 Program Check Pump Commands, CSC-2.1, verifies the Pump Commands library module, CSC-1.3.

3.2.3.2 Program Check Load Commands, CSC-2.2, verifies the Load Commands library module, CSC-1.4.

3.2.3.3 Program Check Test Parameter Routines, CSC-2.3, verifies the Test Parameter library module, CSC-1.5.

3.2.3.4 Program Check Switches, CSC-2.4, verifies the Functions Dumped, CSU-1.3.2, and Switch Set, CSU-1.6.1, and the hardware connections on the counter panel.

3.2.3.5 Program Check Counters, CSC-2.5, verifies the routines associated with the counters in the Test Conditions library module, CSU-1.6.2 through CSU-1.6.6.

3.2.3.6 Program Check Display, CSC-2.6, verifies the sections of the Test Executive, CSC-1.1, concerning the screen, namely Refresh Screen, CSU-1.1.5, Ramp, CSU-1.1.7, Increment Counters, CSU-1.1.8, and Hold, CSU-1.1.9.

3.2.4 CSCI-2 System States and Modes

This section is not applicable to CSCI-2 since all CSCs are stand-alone programs and execute the validation and verification as required.

3.3 CSCI-1 Design Description

This section provides a description of each CSC and CSU for CSCI-1. Figure 3.3 presents the program architecture of the test program in graphical form.

3.3.1 CSC-1.1 Test Executive: This is the main program that executes the test and calls all of the library modules.

3.3.1.1 CSU-1.1.1 Variable Declarations: All variable declarations are included in this CSU as required by Pascal. All constants except those specific to the High Strain Wing Repair program are contained in the Variable Definitions library module, CSC-1.2.

3.3.1.2 CSU-1.1.2 Define Defaults: This CSU sets defaults to reduce the operator interaction upon starting the program. For example, a default is set to run a fatigue test since the majority of the tests will be fatigue. It also opens the fatigue spectrum data file.

3.3.1.3 CSU-1.1.3 Run Menu: This CSU enters the menu state to allow the operator to change the defaults, start the pump or test, or end the program. The operator can elect to change the parameters one-by-one or read a configuration file that contains all of the parameters, which calls CSC-1.5, Test Parameters. The

operator can write a configuration file, which also calls CSC-1.5. The operator can elect to start the pump, at which time CSC-1.3, Pump Commands, is called. The operator can elect to calibrate the load instrumentation, at which time CSC-1.4, Load Commands, is called. Control is passed back to this CSU after any of those operations. The operator can elect to start the test, which passes control to CSU-1.1.4, Initial Load Factors. Lastly, the operator can elect to end the program. If a configuration change has been made that was not previously saved using CSC-1.5, the operator is asked to confirm that he wants the program ended. If the configuration changes were saved or the operator confirms that he wants to exit, control is passed to CSU-1.1.10, Program End.

3.3.1.4 CSU-1.1.4 Initial Load Factors: The load values are initialized to zero, and the counters are reset for the fatigue test. If in the static test mode, the control is passed to CSU-1.1.9, Hold, to allow the operator to enter the next load point. In the fatigue mode, the file pointer is updated and the next load point is determined from the spectrum, the various loading parameters are determined, the screen is updated with the information, and the test is suspended until the operator confirms that he wants the test to be started at that point. If the operator elects to continue, control is passed to CSU-1.1.5, Refresh Screen. If he elects not to continue, control is passed back to CSU-1.1.3, Run Menu. CSC-1.3, Pump Commands, CSC-1.4, Load Commands, and CSC-1.6, Test Conditions, are used in this CSU.

3.3.1.5 CSU-1.1.5 Refresh Screen: This CSU clears the screen, then displays the load information for a static test or spectrum information for a fatigue test, and

displays instructions for the operator on how to halt the test. Control is then passed to CSU-1.1.6, Compute Load Factors.

3.3.1.6 CSU-1.1.6 Compute Load Factors: This CSU uses the load point information to calculate data required for the ramping sequence. See Figure 3.4 for an example of the load point information. Control is then passed to CSU-1.1.7, Ramp.

3.3.1.7 CSU-1.1.7 Ramp: This CSU is where the actual loading of the test element occurs. An incremental load point is determined using a time-based sine function and the load data computed in CSU-1.1.6. The load is incremented and the load feedback is read. At each load increment, a signal is sent to the HP-1000 to trigger a data scan and record the data to the 9-track tape, which calls CSU-1.4.4, Scan-1000. If the load feedback is not within tolerance or the operator has entered a key on the keyboard, control is passed to CSU-1.1.9, Hold. If the pump has dumped, control is passed to CSU-1.1.3, Run Menu. If the load point has been reached, several checks are made. The switches on the counter panel (see Figure 1.1) are checked to see if the operator has turned 'on' the switch to hold at a peak or valley (see Table 3.1), in which case the control is passed to CSU-1.1.9, Hold. If the operator had requested a zero load point, a switch is checked to see if the operator wanted the pump dumped, in which case the pump control is passed to CSU-1.1.3, Run Menu. If neither of these conditions is true, then control is passed to CSU-1.1.8, Increment Counters for a fatigue test, or CSU-1.1.9, Hold, for a static test. CSC-1.3, Pump Commands, CSC-1.4, Load Commands, and CSC-1.6, Test Conditions, are used in this CSU.

3.3.1.8 CSU-1.1.8 Increment Counters: This CSU is only used during a fatigue test. The spectrum pointer is incremented and the load value is read from the spectrum file. The operator switches are checked when the spectrum is at the end of a flight or a pass, in which case control is passed to CSU-1.1.5, Refresh Screen. Otherwise, the load point information is updated, the data displayed on the screen is updated, and the test is continued by passing control to CSU-1.1.6, Compute Load Factors. CSC-1.6, Test Conditions, is used in this CSU.

3.3.1.9 CSU-1.1.9 Hold: Initially this CSU clears the screen, then displays the load information and operator instructions on the screen. The load feedback is read and checked to be sure it is within tolerance. If the load is out of tolerance, then the pump is dumped, and control is passed to CSU-1.1.3, Run Menu. Otherwise, this CSU continues to monitor the load until the operator enters a key. If the operator enters an inappropriate key, it is ignored, and the load is again monitored. If the load point is zero, the operator can elect to return to the main menu, CSU-1.1.3, Run Menu. If the load is not zero, the operator can elect to unload the test specimen, and control is passed to CSU-1.1.7, Ramp. During a fatigue test, the operator can elect to resume the test where it was stopped, and control is passed to CSU-1.1.5, Refresh Screen. During a static test, the operator can elect to trigger a data scan with the HP-1000, which calls CSU-1.4.4, Scan-1000, then resumes monitoring the load. Also during a static test, the operator can elect to go to a new load point. The program will request the value of the load point, check that the load point is within the system capacity, then passes control to CSU-1.1.5, Refresh Screen. CSC-1.3, Pump Commands, CSC-1.4, Load Commands, and CSC-1.6, Test Conditions, are used in this CSU.

3.3.1.10 CSU-1.1.10 Program End: This CSU merely clears the screen, displays 'Program Ended....' on the screen, and exits the program.

3.3.2 CSC-1.2 Variable Definitions (VAR_DEF): This library module holds most of the constants used in the main program and the other modules. The character strings for the titles and menu options are defined. The computer addresses for the slots of the multiprogrammer are defined to simplify the software, for example, the address for slot 401 is 4096 (decimal). The typical locations of the individual cards are defined, for example the relay output card is given as usually located in slot 401. Other definitions include the numerical definition of π , a nominal time increment for calibration routines, and the months of the year. All of the ASCII characters used in write statements to move the cursor, such as backspace, up space, etc., are defined. Note that all other modules will require this module for compiling since they all use these constants. The Pascal *TYPE* identifiers are included to define the string lengths used in the variable declaration statement. Since the purpose of this library module is to define constants, there are no functions or procedures in this module.

3.3.3 CSC-1.3 Pump Commands (PUMP_CMDS): This library module contains the routines to set the relay to power-up and shut-down the hydraulic power supply, and the function to check to see if the hydraulic power supply is running.

3.3.3.1 CSU-1.3.1 Procedure Power-Up (POWER_UP): This procedure performs the steps necessary to zero the load output then close a relay to allow the operator to power-up the hydraulic power supply.

3.3.3.2 CSU-1.3.2 Function Dumped (DUMPED): This function checks a relay that signifies if the pump is running.

3.3.3.3 CSU-1.3.3 Procedure Dump (DUMP): This procedure opens the relay tied to the hydraulic power supply so that it will shut down, then zeros the load output to prevent any possible problems.

3.3.4 CSC-1.4 Load Commands (LOAD_CMDS): This library module contains the procedures to convert the load in pounds to a voltage to be applied to the Servac, to read the feedback voltage and convert that voltage to a load, to continually read the feedback voltage for calibrating the feedback instrumentation, and to send a signal to the HP-1000 to read the load.

3.3.4.1 CSU-1.4.1 Procedure Output Load (OUTPUT_LOAD): This procedure converts the load in pounds to a voltage, then converts the voltage to a decimal representation of the binary word required for the digital-to-analog voltage output card.

3.3.4.2 CSU-1.4.2 Function Load (LOAD): This function reads the A/D card, converts the binary value to a decimal voltage, then converts the voltage to a load in pounds.

3.3.4.3 CSU-1.4.3 Procedure Calibrate Load (CAL_LOAD): This procedure constantly reads the load (using the function Load), and writes the load to the screen until the operator enters a key to stop the operation. This is required so that the operator can fine-tune the load feedback instrumentation.

3.3.4.4 CSU-1.4.4 Procedure Scan-1000 (SCAN_1000): This procedure closes a relay on the Relay Output card which will signal the HP-1000 computer to read the test data. This is used during static tests where the HP-1000 is used to read the strain gages. Two relays are used: the first triggers the scan, the second is set to signify that the desired load point is reached. During this type of test, the nine-track tape drive (part of the HP-1000 computer) records all of the loads and strains at each load increment in the event of a failure occurring before reaching a desired load point. The desired load points are typically in 5 or 10% increments of the test specimen's design limit load. When the desired load point is reached, the HP-1000 is signaled to print the load and strain data on a line printer to be reviewed by the operator. If the loads are not within tolerance, the operator can fine-tune the Servac until the loads are acceptable, at which time the test can proceed.

3.3.5 CSC-1.5 Test Parameters (TEST_PARM): This library module contains the procedures to allow the operator to change the default parameters, and to save the new values for later use. The module also allows the fatigue test to be started at any point in the flight spectrum, and executes the steps required to end the test.

3.3.5.1 CSU-1.5.1 Procedure Change System Parameters (CHNG_SYST_PARAM): This procedure allows the operator to change any of the control system parameters.

3.3.5.2 CSU-1.5.2 Procedure Change Test Parameters (CHNG_TEST_PARAM): This procedure allows the operator to change the loading rate and the load tolerance. The load tolerance is the allowable difference between the load output and the load feedback. If the difference is greater than the load tolerance, the test is halted.

3.3.5.3 CSU-1.5.3 Procedure Number of Passes (NUMBER_PASSES): This procedure allows the operator to input the number of passes of the fatigue test spectrum. The fatigue test will be stopped at the end of the given number.

3.3.5.4 CSU-1.5.4 Procedure Restart (RESTART): This procedure allows the operator to restart the fatigue test at any point in the spectrum.

3.3.5.5 CSU-1.5.5 Procedure Read Configuration File (READ_CONFIG): This procedure allows the operator to read a configuration file with the system parameters, test parameters, and the calibration delay interval. If the program is in the fatigue test mode, the position in the fatigue test spectrum is also recorded. Default file names are provided for the operator to simplify operation, and it is recommended that a different file be used for the static and fatigue test modes since the test parameters are typically different for the two.

3.3.5.6 CSU-1.5.6 Procedure Write Configuration File (WRITE_CONFIG): This procedure writes the configuration file described in 3.3.5.5 above.

3.3.5.7 CSU-1.5.7 Function Program End (PROGRAM_END): This function verifies that the operator wants the program to end, and in addition, if the configuration was changed during the run, verifies whether the operator wants to save the new configuration. This is particularly important when in the fatigue test mode since the position in the fatigue test spectrum had probably changed.

3.3.6 CSC-1.6 Test Conditions (TEST_COND): This library module contains the procedures which check to see if the operator has set any of the switches on the panel during the test. The module also has the procedures necessary to operate the electronic counters (see Figure 1.1). The counters, Waugh Controls Corporation

Minicounter II, Model 428R, are eight digit solid state electronic pulse counters, and have a non-volatile memory when connected to an external battery back-up. The counters are used to indicate the position of the test within the fatigue test spectrum so that recovery can be made in the event of a power loss. The counters are wired to the relay output card of the multiprogrammer. In order to increment the counter, the corresponding relay is closed for approximately five milliseconds, then opened. The counter will increment when the relay is opened (see section 1.5 for more information on the counter). There are also connections to reset the counters to zero. The zero reset connector for the *POINT* counter is wired to the increment connection of the *RECORD* counter, since every *RECORD* starts at *POINT* = 1. Similarly, the zero reset connector for the *RECORD* counter is wired to the increment of the *PASS* counter, since every *PASS* starts at *RECORD* = 1.

3.3.6.1 CSU-1.6.1 Function Switch Set (SWITCH_SET): The on-off switches on the counter panel are tied to the event sense card. This function returns a value of *True* if the switch number passed with the function is *On*. These switches are typically checked at convenient points in the fatigue test spectrum. For example, if the operator wants the test to be halted at the end of a flight, then he would turn switch one *On*. If he wants the test to hold at the peak load to fine-tune the system, then he would turn switch five *On*. The hydraulic power supply is hard wired as switch number six, and is checked with CSU-1.3.2, Function Dumped. See Table 3.1 for the definition of switch positions.

3.3.6.2 CSU-1.6.2 Procedure Close Relay (CLOSE_RELAY): This procedure closes the relay for the counter passed to the procedure. Since the interlock for the hydraulic power supply is hard wired to the seventh relay, this function also ensures

that this relay stays closed. Therefore, two relays are closed with this function, both the relay number passed by the procedure and the hydraulic power supply relay.

3.3.6.3 CSU-1.6.3 Procedure Open Relay (OPEN_RELAY): This procedure opens all relays except for the interlock relay for the hydraulic power supply.

3.3.6.4 CSU-1.6.4 Procedure Zero Counters (ZERO_COUNTERS): This procedure resets all of the counters to zero without opening the interlock relay for the hydraulic power supply.

3.3.6.5 CSU-1.6.5 Procedure Reset Counters (RESET_COUNTERS): This procedure will zero the counters, then increment the counters to the desired point in the fatigue test spectrum.

3.3.6.5 CSU-1.6.6 Procedure Increment Counters (INCR_COUNTER): This procedure increments the desired counter.

3.4 CSCI-2 Design Description

This section provides a description of each CSC and CSU for CSCI-2.

3.4.1 CSC-2.1 Program Check Pump Commands (HSWR_CHECK_PUMP_CMDS):

This program verifies the three CSU's from the Pump Commands library module, CSC-1.3, which is described in 3.3.3 above. This CSC has one CSU that executes the Pump Commands CSU's, the Check Pump Commands Program Executive.

3.4.1.1 CSU-2.1.1 Check Pump Commands Program Executive: This program displays instructions for the operator and the status of the pump by calling CSU-1.3.2, Function Dumped. If the operator enters 'P' to power up the pump, the program calls CSU-1.3.1, Procedure Power-Up. If the operator enters 'D' to dump the pump, the program calls CSU-1.3.3, Procedure Dump. If the operator enters any other key, the program is ended.

3.4.2 CSC-2.2 Program Check Load Commands (HSWR_CHECK_LOAD_CMDS):

This program verifies the four CSU's from the Load Commands library module, CSC-1.4, which is described in 3.3.4 above. This CSC has one CSU that executes the Load Commands CSU's, the Check Load Commands Program Executive. A cable is required to jumper the D-to-A to the A-to-D cards on the multiprogrammer. This cable simulates the feedback signal by sending the output voltage for the load command directly into the card that reads the input voltage.

3.4.2.1 CSU-2.2.1 Check Load Commands Program Executive: This program starts by requesting the operator to enter the maximum test load, in pounds, for the actuator and the full scale voltage for the feedback circuit. After the entry is given, the screen is cleared, the input parameters are displayed, and the operator is

requested to enter the load he would like the computer to ramp to. When the operator enters a load, the program ramps to that load, displaying the command signal and feedback signal as it goes. CSU-1.4.1 Procedure Output Load is called to verify the procedure to output the load. CSU-1.4.2 Function Load is called to verify the procedure to read the load feedback. When the program reaches the load point, the feedback signal is continually updated until the operator enters a new load point, at which time the program again ramps to that load point. In order to change the input parameters, the operator enters '-99' for a load point. The operator can enter the calibration routine, which verifies the operation of CSU-1.4.3, Procedure Calibrate Load. The operator can elect to send a signal to the HP-1000 to trigger a data scan and record the data to the 9-track tape, or to send a signal to the HP-1000 to scan and print the data, which verifies the operation of CSU-1.4.4, Procedure Scan-1000. Lastly, the operator can elect to end the program.

3.4.3 CSC-2.3 Program Check Test Parameter Routines
(HSWR_CHECK_TEST_PARM): This program verifies the seven CSU's from the Test Parameter library module, CSC-1.5, which is described in 3.3.5 above, and the menu section of the test executive, CSU-1.1.3 Run Menu, which is described in 3.3.1.3 above. This CSC has one CSU that executes the Test Parameter library module CSU's, the Check Test Parameter Program Executive.

3.4.3.1 CSU-2.3.1 Check Test Parameter Program Executive: The menu section of the Test Executive (CSU-1.1.3) was utilized for this program executive, and was modified only where required. There are only three significant differences between this CSU and CSU-1.1.3. One, if the operator elects to power up the pump, the

program states that this procedure is verified in the Check Pump Commands program. Two, if the operator elects to calibrate the load reading, the program states that this procedure is verified in the Check Load Commands program. Three, all calls to CSU-1.3.2, Function Dumped, are commented out. Otherwise, the Test Parameter CSU's are verified when the operator selects the appropriate menu item. For example, CSU-1.5.2, Procedure Change Test Parameters, is verified when the operator selects menu item '3', and CSU-1.5.4, Procedure Restart, is verified when the operator selects menu item '9'.

3.4.4 CSC-2.4 Program Check Switches (HSWR_CHECK_SWITCHES): This program verifies the Functions Dumped, CSU-1.3.2, and Switch Set, CSU-1.6.1, which are described in 3.3.3.2 and 3.3.6.1 above, respectively. This program also verifies the hardware connections on the counter panel as described in 3.3.1.7 above. This CSC has one CSU, the Check Switches Program Executive.

3.4.4.1 CSU-2.4.1 Check Switches Program Executive: This program continually displays the status of the six switches and the pump circuit. As the operator changes the state of the switches or pump circuit, the screen should reflect that change. The program ends when the operator enters any key.

3.4.5 CSC-2.5 Program Check Counters (HSWR_COUNTER_CHECK): This program verifies the five CSU's from the Test Conditions library module pertaining to the battery-backup counters on the counter panel, namely CSU-1.6.2 through CSU-1.6.6 described in 3.3.6.2 through 3.3.6.6 above. CSU-1.6.1, Switch Set, is verified in CSC-2.4 described in 3.4.4 above. This program also verifies the hardware connections to the counters. This CSC has two CSU's, Procedure Restart Module and Check Counters Program Executive.

3.4.5.1 CSU-2.5.1 Procedure Restart Module (RESTART_MOD_HSWRCC): The purpose of this procedure is to query the operator for a starting position in the fatigue spectrum, and to pass those values back to the program executive.

3.4.5.2 CSU-2.5.2 Check Counters Program Executive: The program first resets all counters to zero to verify the operation of CSU-1.6.4, Zero Counters. Next, it asks the operator which counter to increment. This checking routine will verify the four counters on the counter panel; however, only three are used for the High Strain Wing Repair fatigue test. CSU-1.6.6, Increment Counter, is verified at this time, which also calls CSU-1.6.2, Close Relay, and CSU-1.6.3, Open Relay. Close Relay and Open Relay are not called in this program executive directly, they are verified based on the call to Increment Counter. If the operator elects to reset all counters to simulate starting a fatigue test in the middle of the spectrum, the program first calls CSU-2.5.1, Restart Module, to determine the starting point, then increments the counters to that point to verify the operation of CSU-1.6.5, Reset Counters. There are several instances where CSU-1.6.1, Switch Set, is called in both the program executive and the CSU's from CSC-1.6. This is to cause a delay to allow the relays to close and the counter to increment. Without these calls, the relays would not remain closed long enough for the counter to recognize the closure. The number of calls is determined through trial and error. A time-based delay could have been used and may have been more efficient but would have required several lines of code. Instead, a dummy variable was created and Switch Set is called which only takes one line of code. The state returned by Switch Set, i.e. *TRUE* or *FALSE*, is not used.

3.4.6 CSC-2.6 Program Check Display (HSWR_CHECK_DISPLAY): This program verifies the CSU's of the Test Executive, CSC-1.1, concerning the CRT screen, namely Refresh Screen, CSU-1.1.5, Ramp, CSU-1.1.7, Increment Counters, CSU-1.1.8, and Hold, CSU-1.1.9. This CSC has one CSU, the Check Display Program Executive.

3.4.6.1 CSU-2.6.1 Check Display Program Executive: This program was taken directly from the appropriate sections of the Test Executive (CSC-1.1), and modified only where required to run as a stand alone program. Those lines of code that were added or commented out are identified to simplify reinstalling the code from this CSU into the test executive. The test executive CSU's are executed in the same order that they are executed in CSCI-1.

Table 3.1 - Counter Panel Switch Definitions

Switch 0	-	Stop at End of Pass
Switch 1	-	Stop at End of Flight
Switch 2	-	Dump Pump After Stopping
Switch 3	-	Not Used
Switch 4	-	Hold at Peak
Switch 5	-	Hold at Valley

- Notes:**
- The switches are applicable only during a fatigue test.
 - Switch 2 is applicable only if Switch 1 or 2 is set.

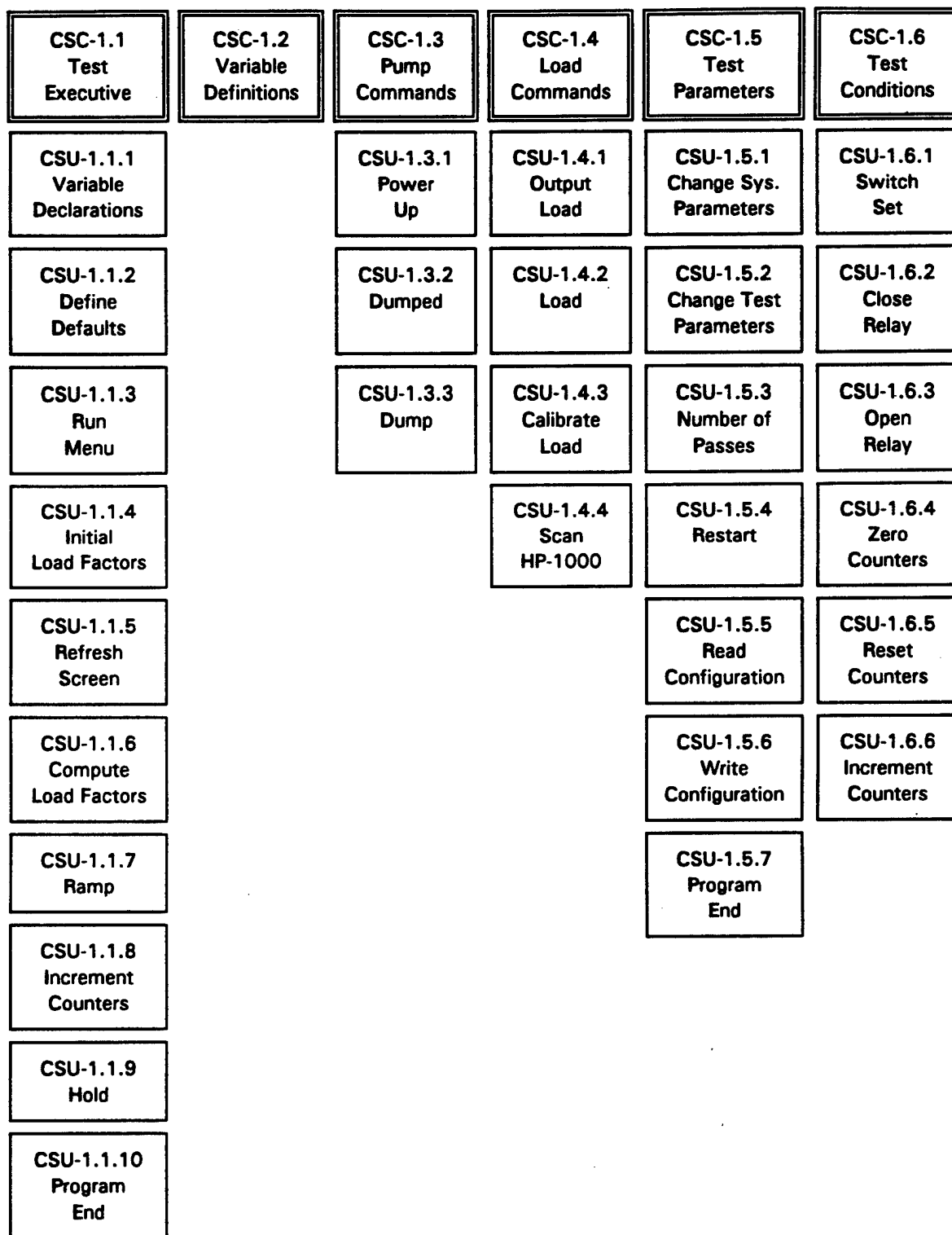


Figure 3.1. CSCI-1 Test Program

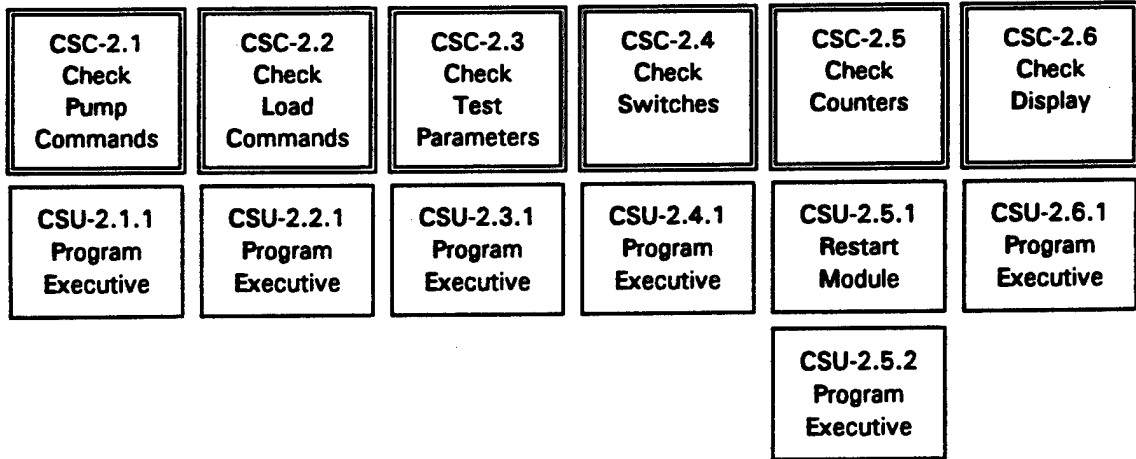


Figure 3.2. CSCI-2 Test Program



Figure 3.3a. Static Test States

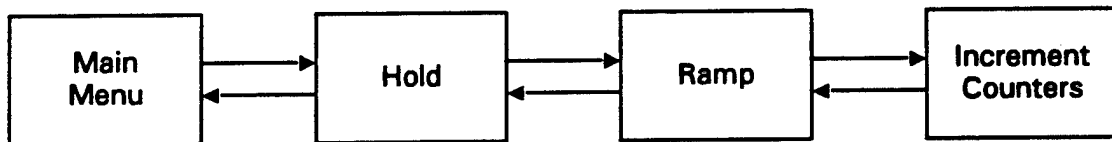


Figure 3.3b. Fatigue Test States

Figure 3.3. CSCI-1 Program Architecture

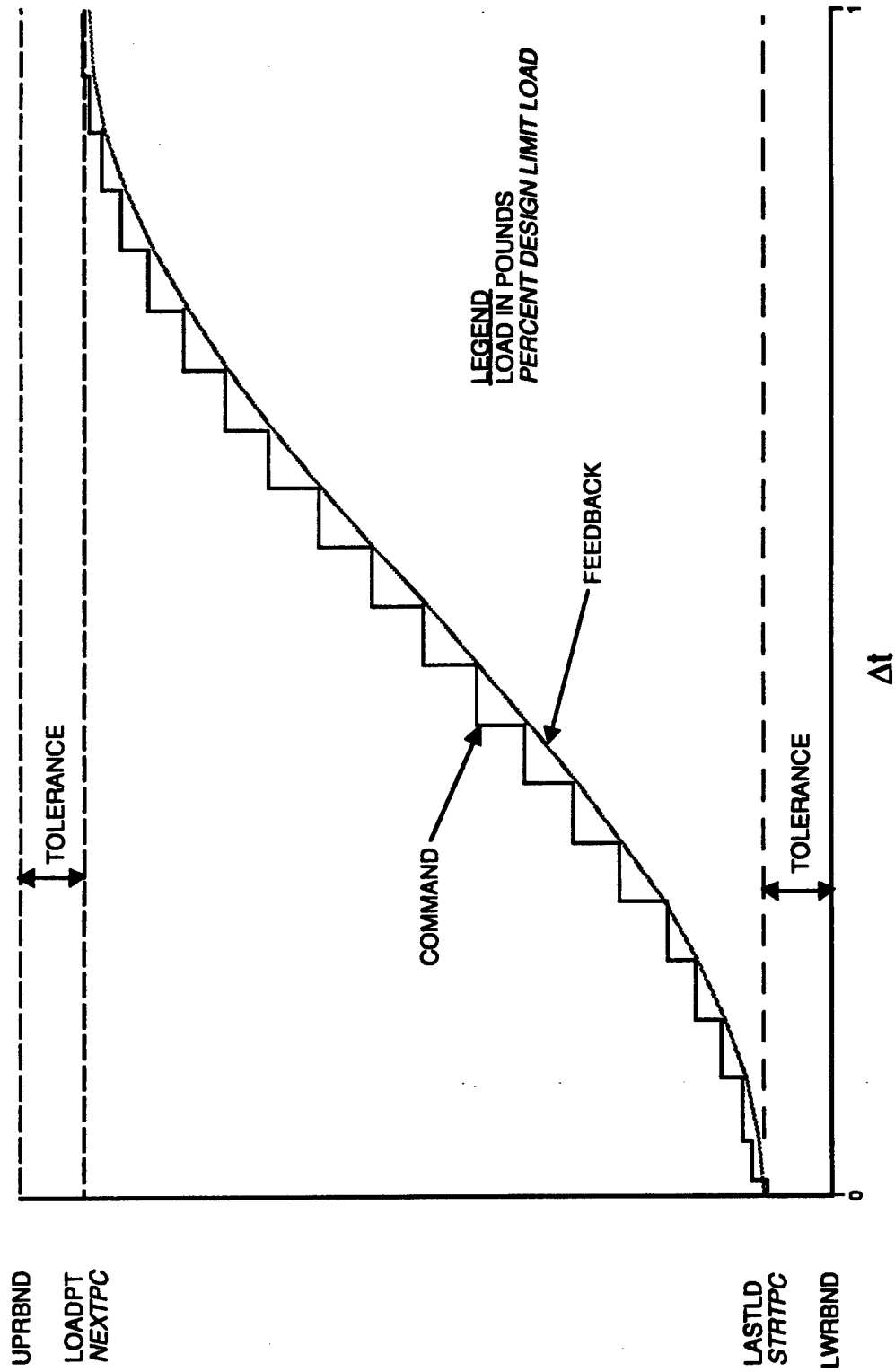


Figure 3.4. Load Point Information

4.0 SOFTWARE USER'S GUIDE

This Software User's Guide provides the instructions necessary to operate the High Strain Wing Repair (HSWR) Test Software and Verification Software.

4.1 Overview

An auto-start routine has been written specifically for the HSWR test. This routine automatically loads the operating system files and the fatigue spectrum, and executes the HSWR test program without any input required from the operator other than swapping floppy disks at the proper time.

The program itself is menu driven to simplify operation. The software driving the menus will not allow an invalid entry by the operator in order to prevent program errors. The software will allow either upper or lower case as a response. For example, a menu that shows 'D' as an acceptable response, will also accept 'd' as well.

When the operator exits the HSWR program, he can initiate the validation routines by following the directions in section 4.3 below. These validation routines are also menu driven, using the same guidelines as mentioned above.

No knowledge of the operating system is required of the operator. If the HSWR program or validation routines require modification, then the operator must have some familiarity with the operating system, editor, compiler, file manager, and system librarian. It is beyond the scope of this document to explain the procedures to modify the programs.

The following floppy are disks associated with the HSWR test:

DAL40 - This disk contains the text and compiled versions of the HSWR program, the configuration files, and the compiled versions of the library modules. This disk is used to edit and run the HSWR program.

DAL41 - This disk contains the text and compiled versions of the library modules. This disk is for editing and compiling the individual library modules.

DAL42 - This disk contains the text and compiled versions of the validation routines. This disk is for developing, editing, compiling, and running the validation routines.

DAL43 - This disk contains the HSWR fatigue spectrum, *WS125*.

DAL44 - This disk is the boot disk with the auto-start file to load the operating system and the fatigue spectrum, and to execute the HSWR program. Note that this disk requires the 3½" floppy disk labeled *HSWRBT* be inserted in the HP-9122 Disk Drive, drive 0. This software will also request the operator to install **DAL40** in the HP-9826 disk drive.

4.2 CSCI-1 Test Software Execution Procedures

4.2.1 Startup Procedures

Turn on the power to the HP-9122 Disk Drive and insert the 3½" floppy disk labeled *HSWRBT* in drive 0. Insert the 5¼" floppy disk labeled **DAL44** into the HP-9826 (there is only one disk drive on the HP-9826) and turn the power on. The

screen will flash several messages as the HP-9826 performs a memory check, loads the operating system, and executes the auto-start routine that will load various files required for the HSWR test software.

Because the required files would not fit on one 3½" and one 5¼" floppy, an additional floppy disk was required. The auto-start routine will halt, and prompt the operator to remove *DAL44* from the HP-9826 disk drive, install *DAL40* in the HP-9826 disk drive, and enter 'X', as shown in Figure 4.1. At this point, the HP-9826 will load the HSWR test program into memory, and start the execution.

4.2.2 Main Menu Procedures

When the program is started, the first screen displayed is the main menu shown Figure 4.2. The program defaults to the fatigue test mode, as indicated by the title at the top of the screen. The program is waiting for the operator to enter a request (Note: the *Return* key is not required). Any key other than the valid numbers shown on the screen will be ignored.

The first available option, '0', toggles the program mode between the fatigue test mode and static test mode, as shown in Figure 4.3. The current mode is always displayed in the title at the top of the screen, and the menu indicates to enter '0' to change to the other mode.

The next option, '1', reads a configuration file. The program will request the name of the configuration file to be read, as shown in Figure 4.4. The program assumes that the required disk is in the HP-9826 floppy disk drive. The operator is only required to enter the filename with no volume label or extension. The program automatically appends the extension *.TX7*. In the static test mode the default file

is *HSWRSCFG*, as shown in Figure 4.4. In the fatigue test mode the default file is *HSWRFCFG*, as shown in Figure 4.5. If the requested filename that is not available or unreadable on the floppy disk, an error statement is displayed as shown in Figure 4.6, and the main menu will be displayed after the operator enters any key. Consult the HP Pascal User's Guide for an explanation of the *IOresult*.

The next option, '2', changes the system parameters. The menu for changing the system parameters is shown in Figure 4.7. To change the design limit load for the test, the operator must enter 'D'. The program will request the operator to enter the new design limit load, as shown in Figure 4.8. After the operator enters the new value, the test ultimate load is recalculated (150% of the design limit load), and the system parameters menu is shown again with the updated values. To change the maximum load for the load control system, the operator must enter 'M'. The program will request the operator to enter the new maximum test load, as shown in Figure 4.9. After the operator enters the new value, the new maximum test load in percent of test ultimate load, the servac span setting, and the load resolution (i.e. the least significant bit in terms of load in pounds) are recalculated, and the system parameters menu is shown again with the updated values. To change the full scale voltage for the feedback system, the operator must enter 'V'. The program will request the operator to enter the new voltage value representing 10 volts full scale, as shown in Figure 4.10. After the operator enters the new value, the resolution is recalculated and the system parameters menu is shown again with the updated values. The operator must enter *Space* to exit the system parameters menu and return to the main menu.

The next option, '3', changes the test parameters. The menu for changing the test parameters is shown in Figure 4.11. To change the loading rate, the operator must enter 'L'. The program will request the operator to enter the new loading rate, as shown in Figure 4.12. To change the load tolerance, the operator must enter 'T'. The program will request the operator to enter the new load tolerance, as shown in Figure 4.13. The operator must enter *Space* to exit the test parameters menu and return to the main menu.

The next option, '4', writes a configuration file. The program will request the name of the configuration file to be written, as shown in Figure 4.14. The program assumes that a disk is already loaded in the HP-9826 floppy disk drive. As with the procedure to read a configuration file, the operator is only required to enter the filename with no volume label or extension; the program automatically appends the extension *.TXT*. In the static test mode the default file is *HSWRSCFG*, as shown in Figure 4.14. In the fatigue test mode the default file is *HSWRFCFG*, as shown in Figure 4.15. If an error results from the request, the error statement is displayed as shown in Figure 4.16, and the main menu will be displayed after the operator enters any key. Consult the HP Pascal User's Guide for an explanation of the *IOresult*.

The next option, '5', sets the relay to allow the hydraulic power supply to be started. Function Dumped (CSU-1.3.2, see section 3.3.3.2) is called every time that the main menu is 'refreshed'. If Function Dumped returns a value of *False*, meaning that the pump is not running, then the menu will write that option '5' is to power up the pump, as shown in Figures 4.2 and 4.3. If Function Dumped returns a value of *True*, the menu will leave the option for '5' blank, as shown in Figures 4.17 and 4.18 for the fatigue and static test modes, respectively. Therefore, any

time a key is entered, even if it is not a valid response, the menu is updated depending on the status of the pump. If the operator starts the pump, yet the menu still shows option '5' is to start the pump as in Figure 4.2 or 4.3, the operator can update the menu by simply entering a *Space* key.

The next option, '6', runs the load calibration routine. The program will continually display the load feedback, as shown in Figure 4.19. To change the delay in updating the feedback, the operator must enter 'D'. The program will request the operator to enter the delay, as shown in Figure 4.20 (note: if the delay is too short, the numbers will change so fast the operator will not be able to read them). Once the operator has entered a new delay, the program will continue with the calibration routine, as shown in Figure 4.19. If the operator enters any key other than 'D', the program will exit the calibration routine and return to the main menu.

The next option, '7', ends the program. First, the program will check if a configuration change has been made or the fatigue spectrum pointer has been changed without saving the changes to a configuration file. If no changes had been made, the program will end, showing the Pascal operating system command line, as shown in Figure 4.21. Otherwise, if the program is in the static test mode, Figure 4.22 is displayed to warn the operator. If the operator enters 'N', the program returns to the main menu. If the operator enters 'Y', the program will end, showing the Pascal operating system command line, as shown in Figure 4.23. If the program is in the fatigue test mode, Figure 4.24 is displayed to warn the operator. Again, if the operator enters 'N', the program returns to the main menu, and if the operator enters 'Y', the program will end, showing the Pascal operating system command line, as shown in Figure 4.25.

The next option, '8', changes the number of passes to be run for a fatigue test. The program will ask the operator to enter the number of passes, as shown in Figure 4.26. After the operator has entered a value, the program will return control to the main menu, and display the value as shown in Figure 4.17. Until either a value is entered by the operator or read from a configuration file, the menu will not show a value for the total number of passes, as shown in Figure 4.2. While in the static test mode, this option is blank, as shown in Figure 4.3.

The next option, '9', starts the test. This option is displayed only when the pump is running, as shown in Figures 4.17 and 4.18 for the fatigue and static test modes, respectively. The program will continue with the static or fatigue test, as described in 4.2.3 and 4.2.3, respectively.

4.2.3 Static Test Procedures

The program will hold at a zero load, as shown in Figure 4.27, and wait for the operator to respond. If the operator enters 'S' for a data scan, a signal is sent to the HP-1000 to read the test data and print the data to the line printer (see 3.3.4.4). The screen on the HP-9826 will not change during this sequence. If the operator enters 'M', the program will return to the main menu (see 4.2.2 for the main menu). This option only appears when the load point is zero. If the operator enters 'L' for the next load point, the program asks the operator to enter the new load in percent of design limit load, as shown in Figure 4.28. If the operator enters an invalid character, the program will ask the operator to try again, as shown in Figure 4.28. If the operator enters a value that is greater than the capacity of the load control system, the program informs the operator and asks the operator to try again, as shown in Figure 4.29.

When a valid load point has been entered, the program updates the screen, as shown in Figure 4.30, and begins ramping to that point. The feedback is continually updated as the load is increased. When the load point is reached, the load is held and the screen is updated, as shown in Figure 4.31. The absolute difference and percent difference is given to assist the operator in fine-tuning the load control system. These values are not shown when the load point is zero since the absolute difference is equal to the feedback in that case. The only other difference between Figures 4.27 and 4.31 is that, because the load point is non-zero, the option to return to the menu is not there, and the operator can enter 'U' to unload to zero. The operator could simply enter 'L' followed by '0' to unload the test article, but the option to unload to zero by entering 'U' is given in the event of a problem to quicken the operator's response time.

If the operator enters a key to halt the test while the program is ramping to a load point, the program will immediately halt the test, redefine the load point as the load at that exact instant, and update the screen as shown in Figure 4.31. For example, if the program is ramping from 10% design limit load to 20%, yet the operator enters a key when the program is at 13.7%, the load point of 20% is ignored, and the load point is defined as 13.7%. The program then follows the same logic as described above for Figure 4.31.

If the load feedback is outside the load tolerances, the test will be halted immediately, the pump is dumped, and the load is quickly ramped to zero. The operator is informed of a positive or negative overload, as shown in Figure 4.32, and the program waits for the operator to respond. After the operator enters any key, the program will return to the main menu (see 4.2.2 for the main menu).

If the pump dumps during the test, the program will be halted and the operator is informed of the condition, as shown in Figure 4.33, then program waits for the operator to respond. After the operator enters any key, the load is ramped to zero and the program will return to the main menu.

Note that when the program is holding, waiting for the operator to respond, the load feedback is checked to be certain that it is within tolerances and the pump is checked to be certain it hasn't dumped. The only time this isn't true is when the program is asking the operator for the next load point.

4.2.4 Fatigue Test Procedures

Initially, the program verifies the position of the fatigue test. The default position is Pass, Flight, and Point = 1, as shown in Figure 4.34. Note that if the operator had read a configuration file with a different fatigue test starting position, that position would be reflected on this screen. At this point, the program is waiting for verification of the position of the fatigue test from the operator. If the operator enters 'N', the operator is queried for the starting point, as shown in Figure 4.35. Note that in this figure, values of 3, 155, and 7 have already been entered by the operator for the pass, flight, and pointer, respectively. After entering the values, the operator again is asked to verify the position of the fatigue test, with the new values displayed, as shown in Figure 4.36. If the operator enters 'N' again, the query cycle is repeated. If the operator enters 'Y', the program continues. If the fatigue test position has been changed, then the electronic counters and the spectrum file pointer must be updated, which can take a several seconds. The program displays notes as these functions are being performed, as shown in Figures 4.37 and 4.38, to inform the operator of the delay.

When the program is ready to start the test, the program displays the fatigue test position, and the corresponding spectrum value in percent of design limit load and in pounds, as shown in Figure 4.39. The corrected value is an 8% increase in the positive loads only. The program also displays the load feedback. The feedback is continually updated until the operator enters a request. If the operator does not want to start at this point, then the operator can enter 'M' to return to the main menu (see 4.2.2 for the main menu). Otherwise, entering any key starts the test.

Once the test is started, the screen is continually updated with the spectrum point, the spectrum value in design limit load and in pounds, and the feedback, as shown in Figure 4.40. At this point, the program will run unattended. The screen is updated to keep the operator informed of the status. Several conditions can stop the test.

The operator can halt the test at any time by entering any key. The load is held at exactly the point the test was stopped, as shown in Figure 4.41. If the operator enters a 'C', the fatigue test will continue from the point it was stopped. If the operator enters a 'U', the program will ramp to zero load, as shown in Figure 4.42. The operator can halt the ramp to zero by entering any key, and the program will again display a screen as shown in Figure 4.41. Upon reaching a zero load, the program holds at zero and waits for an operator response, as shown in Figure 4.43. If the operator enters a 'C', the program will continue to the next point in the fatigue test spectrum. If the operator enters a 'M', the main menu will be displayed (see 4.2.2 for the main menu).

If the load feedback is outside the load tolerances, the test will be halted immediately, the pump is dumped, and the load is quickly ramped to zero. The

operator is informed of a positive or negative overload, as shown in Figure 4.44, and the program waits for the operator to respond. After the operator enters any key, the program will return to the main menu.

If the pump dumps during the test, the program will be halted and the operator is informed of the condition, as shown in Figure 4.45, then program waits for the operator to respond. After the operator enters any key, the load is ramped to zero and the program will return to the main menu.

If the operator sets switch '0' or '1' to *On*, the program will halt at a the end of a pass or flight, respectively. If switch '2' is *Off*, the program will enter the same hold pattern as described above for Figure 4.41. If switch '2' is *On*, the program will ramp the load to zero, dump the pump, and inform the operator, as shown in Figure 4.46, then wait for the operator to respond. After the operator enters any key, the program will return to the main menu. This is useful for running the test unattended, at night for example, but ensuring that the test is stopped at an appropriate interval for inspection or calibration.

If the operator sets switch '5' or '6' to *On*, the program will halt at a peak or valley, respectively. The program will enter the same hold pattern as described above for Figure 4.41. This is useful for checking the accuracy of the load control system.

Lastly, the program will stop when the test reaches the end of the pass corresponding to the number of total passes. The program will be halted, the load ramped to zero, the pump dumped, and the program will return to the main menu.

4.2.5 Restart

After the program has been ended, and control has passed to the Pascal operating system, the operator can restart the test program by entering 'R'. This is still true even if any of the software verification programs, below, have been executed.

4.3 CSCI-2 Verification Software Execution Procedures

The verification software is executed from the Pascal operating system. There are two ways to enter into the Pascal operating system. If the operator is unfamiliar with the Pascal operating system, he can execute the autostart instructions above in section 4.2.1, then end the program in the main menu by entering '7'. If the operator is familiar with the operating system, he can load all of the necessary libraries and execute the verification software directly without using the autostart and running the HSWR test program. In either case, the screen must show the Pascal command line, as shown in Figure 4.47.

4.3.1 Check Pump Commands

To execute the Check Pump Commands Program Executive, the operator must enter an 'X' for 'execute'. The operating system will request the file name. The operator must respond with '#3:HSWRPC', as shown in Figure 4.48. The program will display the status of the pump, and a menu for the operator, as shown in Figure 4.49. If the operator enters a 'P', the relay on the relay output card associated with the pump will be closed. If the operator enters a 'D', the relay will be opened. The default position for the relay when the program is started is *Open*. The status

of the pump is determined from the event sense card. If the relay output card is tied electrically to the event sense card with a 'jumper', the pump status on the screen should indicate *On* when the operator enters 'P', as shown in Figure 4.50, and *Off* when the operator enters 'D', as shown in Figure 4.49. If the operator enters any other key, the program is ended, as shown in Figure 4.51.

4.3.2 Check Load Commands

To execute the Check Load Commands Program Executive, the operator must enter an 'X' for 'execute'. The operating system will request the file name. The operator must respond with '#3:HSWRLC', as shown in Figure 4.52. The program will ask what the maximum test load is in pounds, as shown in Figure 4.53. After the operator enters the maximum test load, the program will ask for the voltage corresponding to 500 kips for the feedback circuit, as shown in Figure 4.54. The program will then display the values entered, the command and feedback (which will be zero upon starting the program), and will wait for the operator to enter a load value to ramp to, as shown in Figure 4.55. As the command signal ramps to the load entered by the operator, the screen is updated with the incremental command signal in pounds and the resulting feedback signal in pounds, as shown in Figure 4.56. In this example, the operator had entered 50,000 pounds for a desired output value. When the program reaches that point, the program will enter a *Hold* mode, and wait for the operator to enter a new value, as shown in Figure 4.57. If the operator enters '-99' as a new value, a menu will be presented to the operator, as shown in Figure 4.58.

While in the menu for the Check Load Commands program, if the operator enters 'M', the program will ask for a new maximum test load, as shown in Figure 4.53.

After the operator has entered the new value, the program will go back to the *Hold* mode. The *Hold* mode is where the program waits for the operator to enter a new output value, as shown in Figure 4.57. If the operator enters 'V', the program will ask for a new voltage corresponding to 500 kips for the feedback circuit, as shown in Figure 4.54. After the operator has entered the new value, the program will go back to the *Hold* mode, as mentioned above. If the operator enters 'C', the program will enter the calibration routine. The program will continually display the load feedback, as shown in Figure 4.59. To change the delay in updating the feedback, the operator must enter 'D'. The program will request the operator to enter the delay, as shown in Figure 4.60 (note: if the delay is too short, the numbers will change so fast the operator will not be able to read them). Once the operator has entered a new delay, the program will continue with the calibration routine, as shown in Figure 4.59. If the operator enters any key other than 'D', the program will exit the calibration routine and return to the *Hold* mode. While in the menu for the Check Load Commands program, if the operator enters 'S' for a data scan, a signal is sent to the HP-1000 to read the test data to nine-track tape (see 3.3.4.4), then the program will go back to the *Hold* mode. If the operator enters 'P', a signal is sent to the HP-1000 to read the test data and print the data to the line printer (see 3.3.4.4), then the program will go back to the *Hold* mode. If the operator enters 'Q', the program will end, as shown in Figure 4.61.

4.3.3 Check Test Parameter Routines

To execute the Check Test Parameter Program Executive, the operator must enter an 'X' for 'execute'. The operating system will request the file name. The operator must respond with '#3:HSWRTP', as shown in Figure 4.62. When the program is

started, the first screen displayed is the main menu shown Figure 4.63. This menu is identical to the one described in 4.2.2, with one exception. The options to power up the pump ('5') and start the test ('9') are both shown in this program, as shown in Figures 4.63 and 4.64, while the Test Executive will only show one or the other, as shown in Figures 4.2 and 4.3.

The program is waiting for the operator to enter a request. Any key other than the valid numbers shown on the screen will be ignored. The options '0', '1', '2', '3', '4', '7', '8', and '9', are executed as discussed in 4.2.2 above. If the operator enters '5' to power up the pump, the program will display that this program is for checking the routines in the TEST_PARM library module and the pump commands are verified in the HSWRPC program (described in 4.3.1 above), as shown in Figure 4.65. When the operator enters any key, the program returns to the main menu, and the key that was entered is ignored by the program. If the operator enters '6' to calibrate the load reading, the program will display that the load command routines are verified in the HSWRLC program (described in 4.3.2 above), as shown in Figure 4.66. Again, when the operator enters any key, the program returns to the main menu, and the key that was entered is ignored by the program.

4.3.4 Check Switches

To execute the Check Switches Program Executive, the operator must enter an 'X' for 'execute'. The operating system will request the file name. The operator must respond with '#3:HSWRSC', as shown in Figure 4.67. The program will display the status of the six switches and the pump. Note that the status of the pump is displayed since this is effectively the seventh switch on the Event Sense card, so this function is verified in both this program executive and in the Check Pump

Commands program (see 4.3.1). The switches are checked continually, and the screen will change as the state of the switches changes. For example, all switches and the pump are *Off* in Figure 4.68. All switches are *On* in Figure 4.69. Switches '0', '1', '4', and the pump are *On*, and switches '2', '3', and '5' are *Off* in Figure 4.70. If the operator enters any key, the program is ended, as shown in Figure 4.71.

4.3.5 Check Counters

To execute the Check Counters Program Executive, the operator must enter an 'X' for 'execute'. The operating system will request the file name. The operator must respond with '#3:HSWRCC', as shown in Figure 4.72. The program will display a menu for the operator and the current value for the counters, as shown in Figure 4.73. Note that this checking routine will verify the four counters on the counter panel; however, only three are used for the HSWR fatigue test.

If the operator enters '1', '2', '3', or '4', the program will query the operator for how many times the counter should be incremented. For example, by entering '1', the program will ask how many times to increment counter 1, as shown in Figure 4.74. If the operator enters '5' to simulate starting the HSWR fatigue test, the program will query the operator to enter the desired pass, record, and flight numbers. For example, Figure 4.75 shows operator inputs of pass = 3, record = 25, and flight = 77. After these values are entered, the program will query the operator to be certain that he wants to restart the test at that point, as shown in Figure 4.76. If the operator enters a 'Y', the counters are incremented, and the counters and the screen, shown in Figure 4.77, will give the same values. If the operator enters a 'N', the counters are not incremented, and the screen will reflect

the same values prior to the operator entering '5' from the menu. If the operator enters a 'Q', the program will end, as shown in Figure 4.78.

4.3.6 Check Display

To execute the Check Display Program Executive, the operator must enter an 'X' for 'execute'. The operating system will request the file name. The operator must respond with '#3:HSWRDS', as shown in Figure 4.79. The program will ask the operator which test mode to run, static or fatigue, as shown in Figure 4.80. The mode cannot be changed once it has been entered. If the operator wishes to run the other mode, the program must be ended and re-run.

If the operator enters 'S' for the static test mode, the same procedures will be executed as discussed in 3.2.3 above, with the following exceptions. The only functions executed by this program are those that relate to updating the screen. The load point, command signal, and feedback are displayed on the screen as they are during the execution of the Test Executive (CSC-1.1), but the load output (voltage to the load control system) will be held at zero, and the feedback will not be checked to determine if it is within tolerance. The 'S' option, trigger a data scan, will not actually send the signal to the HP-1000. When the operator enters 'L' and a new load point, this program will not actually ramp to the load point. To change from the ramp function to the hold function requires the operator to enter any key. Also, when at a zero load point, if the operator enters 'M', this program will be ended, as shown in Figure 4.81.

If the operator enters 'F' for the fatigue test mode, the same procedures will be executed as discussed in 3.2.4 above, with the following exceptions. As with the

static test mode, the only functions executed by this program are those that relate to updating the screen. This program will step through a fictitious fatigue spectrum of ten data points, one point every second, but will hold the load output to zero. If the operator enters a key, he will be given the same options as discussed in 3.2.4. To end the program, the operator must halt the test at a zero load point, and enter 'M'. The program will respond as shown in Figure 4.82.

Please put DAL40 in unit #3
and press the X key...

Figure 4.1. Autostart Request for Disk Labeled *DAL40*

HIGH STRAIN WING REPAIR FATIGUE TEST

0. Switch to Static Test
1. Read configuration files
2. Change System parameters
3. Change Test parameters
4. Write configuration files
5. Power up the pump
6. Calibrate load reading
7. End program
8. Change number of passes

Request?

Figure 4.2. Fatigue Test Main Menu

HIGH STRAIN WING REPAIR STATIC TEST

0. Switch to Fatigue Test
1. Read configuration files
2. Change System parameters
3. Change Test parameters
4. Write configuration files
5. Power up the pump
6. Calibrate load reading
7. End program
- 8.

Request?

Figure 4.3. Static Test Main Menu

What is file name to be read?
Default is 'HSWRSCFG' -

Figure 4.4. Request for Static Test Configuration File Name to Read

What is file name to be read?
Default is 'HSWRFCFG' -

Figure 4.5. Request for Fatigue Test Configuration File Name to Read

What is file name to be read?

Default is 'HSWRFCFG' -

Can't open file #3:HSWRFCFG.TEXT.

IOresult= 34

Hit any key to continue....

Figure 4.6. Example of Error While Reading Configuration File

Design Limit load is 167200 lbs
Test Ultimate load is 250800 lbs

Control System:

Maximum Load is 86%TUL
Maximum Load is 215000 lbs
Set Servac Span to 4.30
Resolution is 108 lbs

Feedback System:

Full scale for 500 kips is 10.000 volts
Resolution is 250 lbs

Hit:

D to change Design limit load
M to change Maximum actuator load
V to change full scale Voltage
<space> to exit

Figure 4.7. Menu for Changing the System Parameters

Enter Design Limit Load in pounds -

Figure 4.8. Query to Operator for the Design Limit Load

Enter the maximum test load in pounds -

Figure 4.9. Query to Operator for the Maximum Test Load

What is corresponding voltage for 500kips?

Figure 4.10. Query to Operator for the Full Scale Voltage

Loading Rate is 800 lbs/sec

Tolerance is 5000 lbs

Hit:

L to change Loading rate
T to change Tolerance
<space> to exit

Figure 4.11. Menu for Changing the Test Parameters

What is the loading rate in lbs/sec?

Figure 4.12. Query to Operator for the Loading Rate

Enter tolerance in pounds -

Figure 4.13. Query to Operator for the Load Tolerance

What is file name to be written?
Default is 'HSWRSCFG' -

Figure 4.14. Request for Static Test Configuration File Name to Write

What is file name to be written?
Default is 'HSWRFCFG' -

Figure 4.15. Request for Fatigue Test Configuration File Name to Write

What is file name to be written?
Default is 'HSWRSCFG' -

Can't open file #3:HSWRSCFG.TEXT,
IOresult= 34
Hit any key to continue....

Figure 4.16. Example of Error While Writing Configuration File

HIGH STRAIN WING REPAIR FATIGUE TEST

0. Switch to Static Test
 1. Read configuration files
 2. Change System parameters
 3. Change Test parameters
 4. Write configuration files
 - 5.
 6. Calibrate load reading
 7. End program
 8. Change number of passes, currently 100
 9. Begin test
- Request?

Figure 4.17. Fatigue Test Main Menu with Pump Running

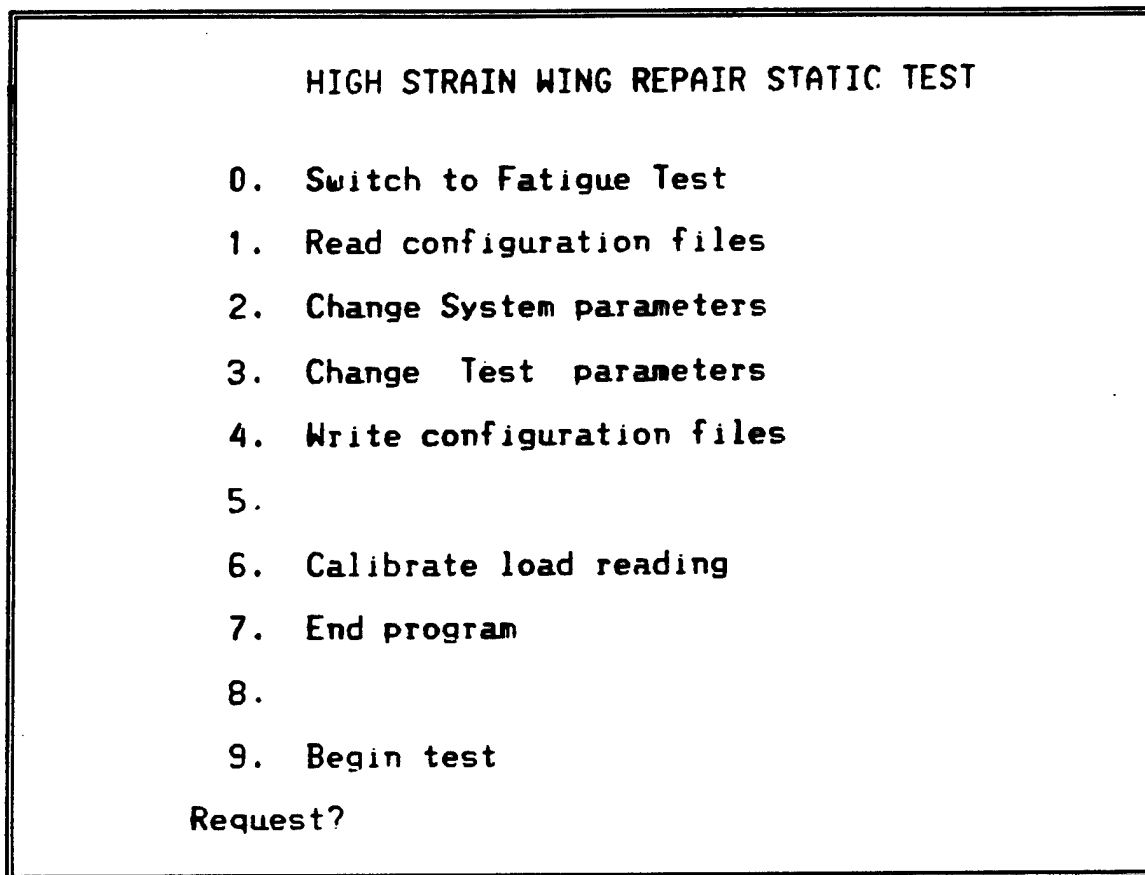


Figure 4.18. Static Test Main Menu with Pump Running

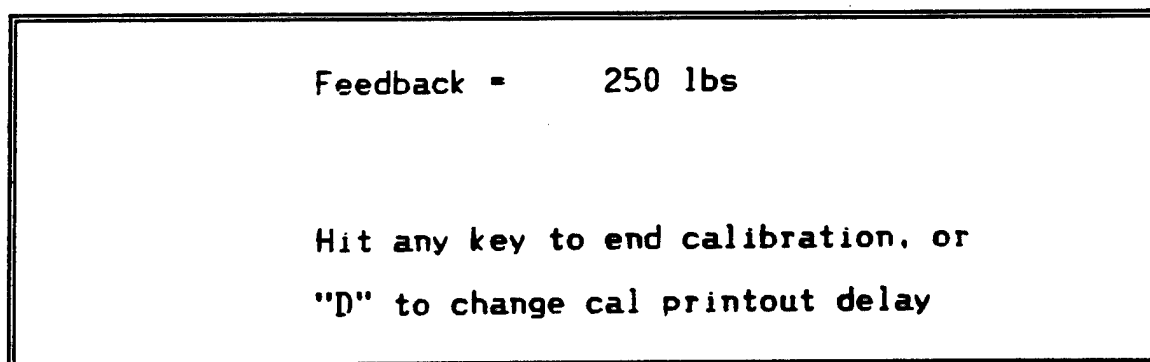


Figure 4.19. Load Calibration Display

Current delay is 100 milliseconds
What is desired delay in milliseconds?

Figure 4.20. Query to Operator for the Calibration Delay

Command: Cmplr Edit File Init Libr Run Xcut Ver ?

Program Ended....

Figure 4.21. Program Ended with No Configuration Changes

Are you sure you want to end the program?
Configuration changes will be lost....

End the program(Y or N)?

Figure 4.22. Configuration Change Notice During a Static Test

Command: Cmplr Edit File Init Libr Run Xcut Ver ?

Are you sure you want to end the program?
Configuration changes will be lost....

End the program(Y or N)?

Program Ended....

Figure 4.23. Static Test Ended with Configuration Changes Not Saved

Are you sure you want to end the program?
Configuration changes and/or the fatigue
test counters will be lost....

End the program(Y or N)?

Figure 4.24. Configuration Change Notice During a Fatigue Test

```
Command: Cmplr Edit File Init Libr Run Xcut Ver ?

Are you sure you want to end the program?
Configuration changes and/or the fatigue
test counters will be lost....

End the program(Y or N)?

Program Ended....
```

Figure 4.25. Fatigue Test Ended with Configuration Changes Not Saved

```
How many passes to be run?
```

Figure 4.26. Query to Operator for the Number of Passes to Run

HIGH STRAIN WING REPAIR STATIC TEST

Holding at 0.0% of DLL
 0.0% of TUL

Desired Load = 0 lbs
 Feedback = 0 lbs

Hit:
 S to trigger data Scan
 L for next Load point
 M to return to the Menu

Figure 4.27. Hold at Zero Load During Static Test

HIGH STRAIN WING REPAIR STATIC TEST

Holding at 0.0% of DLL
 0.0% of TUL

Desired Load = 0 lbs
 Feedback = 0 lbs

What is new load point in percent of DLL?

Figure 4.28. Query to Operator for the Next Load Point

HIGH STRAIN WING REPAIR STATIC TEST

Holding at 110.0% of DLL

73.3% of TUL

Desired Load = 183920 lbs

Feedback = 8555 lbs

Absolute Difference = -175365 lbs

Percent Difference = 95.3 %

What is new load point in percent of DLL? 160

Maximum load is +/-215000lbs (129% DLL)

Try again, new load point?

Figure 4.29. Example of Invalid Request for Next Load Point

HIGH STRAIN WING REPAIR STATIC TEST

Ramping to 10.0% of DLL

6.7% of TUL

Desired Load = 16720 lbs

Feedback = 8000 lbs

Hit any key to halt test....

Figure 4.30. Ramp to Load Point During Static Test

HIGH STRAIN WING REPAIR STATIC TEST

Holding at 10.0% of DLL

6.7% of TUL

Desired Load = 16720 lbs

Feedback = 16250 lbs

Absolute Difference = -470 lbs

Percent Difference = 2.8 %

Hit:

S to trigger data Scan

L for next Load point

U to Unload

Figure 4.31. Hold at Load Point During Static Test

HIGH STRAIN WING REPAIR STATIC TEST

Ramping to 80.0% of DLL

53.3% of TUL

Desired Load = 133760 lbs

Feedback = -67000 lbs

Test stopped on Negative overload....

Lower bound = -66560lbs

Hit any key to continue....

Figure 4.32. Example of Overload During a Static Test

HIGH STRAIN WING REPAIR STATIC TEST

Ramping to 30.0% of DLL

20.0% of TIJL

Desired Load = 50160 lbs

Feedback = 2060 lbs

External dump at 26.52% of DLL

Hit any key to zero command signal
and continue....

Figure 4.33. Example of External Dump During a Static Test

Pass Number = 1

Flight Number = 1

Point Number = 1

Restart at this point? (Y or N) -

Figure 4.34. Fatigue Test Starting Position Verification

Enter the Pass number
between 1 and 100 - 3

Enter the Flight number
between 1 and 355 - 155

Enter the Point number
between 1 and 108 - 7

Figure 4.35. Example of Operator Entry of New Starting Position

Pass Number = 3
Flight Number = 155
Point Number = 7
Restart at this point? (Y or N) -

Figure 4.36. Verification of Operator Entries

Resetting counters. please wait....

Figure 4.37. Note to Operator That Counters are Being Updated

Setting file pointer. please wait....

Figure 4.38. Note to Operator That File Pointer is Being Updated

HIGH STRAIN WING REPAIR FATIGUE TEST

Pass Number =	3
Flight Number =	155
Point Number =	7
Spectrum Value =	-14.0 %DLL
Corrected Value =	-14.0 %DLL
Desired Load =	-23408 lbs
Feedback =	0 lbs

Hit any key to start test, or
M to return to menu....

Figure 4.39. Final Verification Prior to Starting Test

HIGH STRAIN WING REPAIR FATIGUE TEST

Pass Number = 3
Flight Number = 155
Point Number = 7
Spectrum Value = -14.0 %DLL
Corrected Value = -14.0 %DLL
Desired Load = -23408 lbs
Feedback = -12750 lbs

Hit any key to halt test....

Figure 4.40. Example of Display During Fatigue Test

HIGH STRAIN WING REPAIR FATIGUE TEST

Holding at -28.9% of DLL
-19.2% of TUL

Desired Load = -48269 lbs
Feedback = -48750 lbs
Absolute Difference = -481 lbs
Percent Difference = 1.0 %

Hit:
C to Continue test were stopped
U to Unload

Figure 4.41. Example of Operator Interrupt During Fatigue Test

HIGH STRAIN WING REPAIR FATIGUE TEST

Ramping to 0.0% of DLL
0.0% of TUL

Desired Load = 0 lbs

Feedback = -7750 lbs

Hit any key to halt test....

Figure 4.42. Ramp to Zero Load Initiated by Operator Request

HIGH STRAIN WING REPAIR FATIGUE TEST

Holding at 0.0% of DLL
0.0% of TUL

Desired Load = 0 lbs

Feedback = 0 lbs

Hit:

C to Continue test were stopped
M to return to the Menu

Figure 4.43. Hold at Zero Load During Fatigue Test

HIGH STRAIN WING REPAIR FATIGUE TEST

Pass Number = 3
Flight Number = 155
Point Number = 7
Spectrum Value = -14.0 %DLL
Corrected Value = -14.0 %DLL
Desired Load = -23408 lbs
Feedback = -28750 lbs

Test stopped on Negative overload....
Lower bound = -28408lbs
Hit any key to continue....

Figure 4.44. Example of Overload During a Fatigue Test

```

HIGH STRAIN WING REPAIR FATIGUE TEST

Pass Number =      1
Flight Number =     1
Point Number =     2
Spectrum Value =  -55.3 %DLL
Corrected Value =  -55.3 %DLL
Desired Load =  -92462 lbs
Feedback =      500 lbs

External dump at -4.84% of DLL
Hit any key to zero command signal
and continue....
    
```

Figure 4.45. Example of External Dump During a Fatigue Test

```

Test stopped and pump dumped
per operator request.

Hit any key to continue....
    
```

Figure 4.46. Test Stopped Using Counter Panel Switches

```

Command: Cmplr Edit File Init Libr Run Xcut Ver ?
    
```

Figure 4.47. Pascal Operating System Command Line

Execute what file? #3:HSWRPC

Figure 4.48. Operator Input to Execute the Check Pump Commands Program

Pump is Off

Hit any key to quit, or
"P" to Power up pump
"D" to Dump pump

Figure 4.49. The Check Pump Commands Program Menu with Pump *Off*

Pump is On

Hit any key to quit, or
"P" to Power up pump
"D" to Dump pump

Figure 4.50. The Check Pump Commands Program Menu with Pump *On*

Command: Cmplr Edit File Init Libr Run Xcut Ver ?

Pump is On

Program ended....

Figure 4.51. Display After the Check Pump Commands Program Has Ended

Execute what file? #3:HSWRLC

Figure 4.52. Operator Input to Execute the Check Load Commands Program

Enter the maximum test load in pounds -

Figure 4.53. Query to Operator for the Maximum Test Load

What is corresponding voltage for 500 kips?

Figure 4.54. Query to Operator for the Full Scale Voltage

Command: Maximum Load is 500000 lbs

Feedback: Full scale for 500 kips is 10.000 volts

Command = 0 lbs

Feedback = 0 lbs

Enter Output value in pounds.

or -99 to stop

Figure 4.55. Query to Operator for the Output Load

Command: Maximum Load is 500000 lbs

Feedback: Full scale for 500 kips is 10.000 volts

Command = 10179 lbs

Feedback = 10250 lbs

Enter Output value in pounds.

or -99 to stop

Figure 4.56. Example of Display During Ramp to Output Load

Command: Maximum Load is 500000 lbs

Feedback: Full scale for 500 kips is 10.000 volts

Command = 50000 lbs

Feedback = 50000 lbs

Enter Output value in pounds.

or -99 to stop

Figure 4.57. Query to Operator for the Next Output Load While Holding

Command: Maximum Load is 500000 lbs

Feedback: Full scale for 500 kips is 10.000 volts

Command = 100000 lbs

Feedback = 100000 lbs

What would you like to do now? Hit:

M to change Maximum actuator load

V to change full scale Voltage

C to run Calibration routine

S to sent Scan signal to HP-1000

P to sent Print signal to HP-1000

Q to Quit

Figure 4.58. Menu for Changing the Load Parameters

Feedback = 0 lbs

Hit any key to end calibration, or
"D" to change cal printout delay

Figure 4.59. Load Calibration Display

Current delay is 100 milliseconds
What is desired delay in milliseconds?

Figure 4.60. Query to Operator for the Calibration Delay

Command: Cmplr Edit File Init Libr Run Xcut Ver ?
 Command: Maximum Load is 500000 lbs
 Feedback: Full scale for 500 kips is 10.000 volts

Command = 10000 lbs
 Feedback = 10000 lbs

What would you like to do now? Hit:

M to change Maximum actuator load
 V to change full scale Voltage
 C to run Calibration routine
 S to sent Scan signal to HP-1000
 P to sent Print signal to HP-1000
 Q to Quit

Program ended....

Figure 4.61. Display After the Check Load Commands Program Has Ended

Execute what file? #3:HSWRTP

Figure 4.62. Operator Input to Execute the Check Test Parameter Program

HIGH STRAIN WING REPAIR, FATIGUE TEST	
0.	Switch to Static Test
1.	Read configuration files
2.	Change System parameters
3.	Change Test parameters
4.	Write configuration files
5.	Power up the pump
6.	Calibrate load reading
7.	End program
8.	Change number of passes
9.	Begin test
Request?	

Figure 4.63. Check Test Parameter Program Fatigue Test Menu

HIGH STRAIN WING REPAIR STATIC TEST

0. Switch to Fatigue Test
1. Read configuration files
2. Change System parameters
3. Change Test parameters
4. Write configuration files
5. Power up the pump
6. Calibrate load reading
7. End program
- 8.
9. Begin test

Request?

Figure 4.64. Check Test Parameter Program Static Test Menu

This program is for checking the routines
in TEST_PARM. Use HSWRPC for checking the
pump command routines.

Hit any key to continue....

Figure 4.65. Check Test Parameter Program Response to Menu Item '5'

This program is for checking the routines
in TEST_PARM. Use HSWRLC for checking the
load command routines.

Hit any key to continue....

Figure 4.66. Check Test Parameter Program Response to Menu Item '6'

Execute what file? #3:HSWRSC

Figure 4.67. Operator Input to Execute the Check Switches Program

Switch Number						
0	1	2	3	4	5	Pump
On	Off	Off	Off	Off	Off	Off
Hit any key to quit....						

Figure 4.68. Example of Display With All Switches and Pump *Off*

Switch Number						
0	1	2	3	4	5	Pump
On	On	On	On	On	On	On
Hit any key to quit....						

Figure 4.69. Example of Display With All Switches *On*

Switch Number						
0	1	2	3	4	5	Pump
On	On	Off	Off	On	Off	On
Hit any key to quit....						

Figure 4.70. Example of Display With Switches and Pump Randomly *On* and *Off*

```
Command: Cmplr Edit File Init Libr Run Xcut Ver ?

                                Switch Number

      0      1      2      3      4      5      Pump
Off Off Off Off Off Off Off
Hit any key to quit....

Program ended....
```

Figure 4.71. Display After the Check Switches Program Has Ended

```
Execute what file? #3:HSWRCC
```

Figure 4.72. Operator Input to Execute the Check Counters Program

Enter counter number you want changed.
'5' to change 1, 2, and 3 all at the same
time (like restarting a fatigue test).
or 'Q' to quit....

Counter 1 = 0
2 = 0
3 = 0
4 = 0

Figure 4.73. Check Counters Program Menu at Start of Program

How many times do you
want counter 1 incremented?

Figure 4.74. Query to Operator for the Number of Times to Increment a Counter

Enter the Pass number - 3
Enter the Record number - 25
Enter the Point number - 77

Figure 4.75. Example of Operator Entry of New Starting Position

Pass Number = 3
 Record Number = 25
 Point Number = 77
 Restart at this point? (Y or N) -

Figure 4.76. Verification of Operator Entries

Enter counter number you want changed,
 '5' to change 1, 2, and 3 all at the same
 time (like restarting a fatigue test),
 or 'Q' to quit....

Counter 1 = 77
 2 = 25
 3 = 3
 4 = 0

Figure 4.77. Check Counters Program Menu after Incrementing Counters

Command: Cmplr Edit File Init Libr Run Xcut Ver ?

Enter counter number you want changed.

'5' to change 1, 2, and 3 all at the same
time (like restarting a fatigue test).

or 'Q' to quit....

Counter	1	=	77
	2	=	25
	3	=	3
	4	=	0

Program ended....

Figure 4.78. Display After the Check Counters Program Has Ended

Execute what file? #3:HSWRDS

Figure 4.79. Operator Input to Execute the Check Display Program

Static or Fatigue (S or F)?

Figure 4.80. Query to Operator for the Test Mode

Command: Cmplr Edit File Init Libr Run Xcut Ver ?

Holding at 0.0% of DLL

0.0% of TUL

Desired Load = 0 lbs

Feedback = 0 lbs

Hit:

S to trigger data Scan

L for next Load point

M to return to the Menu

Program Ended....

Figure 4.81. Display After the Check Display Program Has Ended in the Static Mode

Command: Cmplr Edit File Init Libr Run Xcut Ver ?

Holding at 0.0% of DLL

0.0% of TUL

Desired Load = 0 lbs

Feedback = 0 lbs

Hit:

C to Continue test were stopped

M to return to the Menu

Program Ended....

Figure 4.82. Display After the Check Display Program Has Ended in the Fatigue Mode

APPENDIX A - HARDWARE DESCRIPTION

This section gives a description of the hardware associated with structural test. This includes both the computer hardware and the loading apparatus.

A.1 Description of test setup

On a typical test setup, loads are imparted on the test specimen or subcomponent by a hydraulic actuator. On the test floor, one or more actuators are mounted in a custom designed fixture to match the loading parameters desired for the test specimen or subcomponent, and reactions for statically indeterminate applications are instrumented by load cells. For example, a beam in three-point loading requires only one load cell on the hydraulic actuator, but a subcomponent that simulates a bulkhead attachment to a fuselage will have several actuators to impart shear and torsion on the subcomponent, so the reactions are typically instrumented with a load cell. In the fatigue lab, the hydraulic actuator is permanently mounted in a load frame. Interchangeable grips are used for clamping a test specimen into the load frame. There are six load frames, or more commonly referred to as test machines; three 20-kip capacity and three 100-kip capacity.

In all cases, power to the hydraulic actuators is supplied by one of three hydraulic power supplies or pumps, which operate at 3000psi. The pumps are operated from a control panel. Relays in the control panel act as safety interlocks. When all of the relays are closed, the control panel allows an operator to start the pump. When any of the relays are opened, the control panel shuts off the pump. The relays are tied in to any of a number of different sensors, some of which are on the pump

itself. For example, there is a relay tied to a pump over-temperature indicator, and a relay tied to a low oil level indicator.

Some of the relays are tied to switches on the hydraulic actuators, which are stroke limits. These stroke limits are set manually by the test engineer or test mechanics, and prevent a runaway condition where the hydraulic actuator may move further than anticipated.

A.2 Description of test equipment

The actuators are servo-controlled hydraulic actuators. The load output of the actuator is dependent on the hydraulic pressure applied to the piston in the actuator. A small servo mounted on the side of the actuator regulates the amount of hydraulic pressure applied to the piston. The servo is an electromagnetic device. The servo valve is controlled electrically by an MTS Model 401.03 Servac, which will be described below.

A load cell attached to actuator translates axial load into electrical resistance by means of a four-way bridge. Each load cell contains two electrically independent bridges, the "A" bridge and the "B" bridge. The "B" bridge is typically tied to some sort of load reading device, as required. Generally in the fatigue lab, the "B" bridge is tied to a voltmeter that converts load to voltage directly. A computer or oscilloscope can also be tied into the "B" bridge. Generally on the test floor, load is read by a computer.

The servac is the interface between the operator and the hydraulic actuator. The operator applies an input voltage to the servac depending on the desired load. The

actual calculation of the input voltage from the desired load in pounds is dependent on the size of the actuator, the load cell.

The servac applies a voltage to one leg of the "A" bridge, and reads the resulting voltage, which is dependent on the load applied to the load cell. This voltage is referred to as the feedback voltage. The Servac compares the input voltage to the feedback voltage, and changes the command voltage to open or close the servo valve as required. The difference between the command voltage and the feedback voltage is known as the error signal. When the error signal becomes too great, the servac opens one of the relays in the control panel to shut the pump down. The operator can adjust the allowable error to suit the particular loading characteristics of the test specimen and loading conditions.

A.3 Description of existing fatigue lab system

There are several methods available to the operator to control the loading of a test machine in the fatigue lab. The function generator is used often. The function generator has a variety of functions to tailor the load control. For example, the operator can select a straight line ramp from zero to +10 volts or from zero to -10 volts to run a tension or compression static test, respectively. The function generator also has a sine wave function for the same purpose. The function generator can also control constant amplitude fatigue. The operator can select a sine wave function that will perform a tension-tension or tension-compression fatigue test. The function generator outputs ± 10 volts as the peak command, the set point on the servac determines the valley load, and the span setting on the servac sets the amplitude of the load.

A.4 Description of existing HP-1000 system

The test floor is much different from the fatigue lab in that no simple tests are conducted on the test floor. The HP-1000 has several interfaces for outputting different loads and reading many different parameters, such as load, strain, deflection, temperature, etc.

- 8 channels D/A for outputting load
- 24 channels A/D for reading load
- 288 channels low level A/D for reading strain & temperature
- 32 relay output channels for controlling machines
- 2 event sense channels for checking status of machines
- 1.5 Megabyte storage RAM
- 20 megabyte storage on disk
- 1 9-track tape drive

There are several advantages of the HP-1000 based test system.

- The test can have as many different controlling parameters as the eight digital to analog converters will provide; many tests have symmetry to allow several actuators to be tied to one output channel.
- Speed is another advantage of this system. The analog to digital converters are extremely fast for reading data, regardless of the number of data channels. Typically, throughput is a function of the number of safety checks required in a complex test, not the data acquisition.

- The data storage capability is nearly unlimited. In addition to the disk drive, the 9-track tape drive gives increased flexibility for storing large quantities of data.

However, there are several disadvantages to the HP-1000 based system.

- There is only one of them, and the cost of an equivalent system was too much. The computer could not be time-shared, since tests are usually performed on one-of-a-kind subcomponents that have taken many years and millions of dollars to design and produce that a dedicated computer system is required to oversee testing. Therefore, while one test is being performed, data processing or software development for another test cannot be conducted. Also, if the computer should go down, there is no backup system to perform time-critical testing.
- There is no time based generator available, so all loading must be done in either a straight ramp or haversine. Most testing is done with haversine, which is a sine wave broken down into a set number of discrete steps, typically 48, sometimes 24 or 96. When a fast loading rate is required, haversine loading can cause distinctly noticeable 'chugging' in the actuators. In some cases this can be the limiting factor for the speed of the fatigue test. A more desirable loading method is a smooth, continuous function.
- The HP-1000 system has very few event sense channels, which can limit the operators flexibility.

A.5 Description of HP-9826 system

Several HP-9826 computers were purchased to perform a scaled down function of the larger HP-1000 system. The HP-9826 is a desktop computer, with a minimum of 256 kilobytes and a maximum of 1 megabyte of RAM, and a 256 kilobytes floppy disk. The HP-9826 has both a Basic and Pascal operating system developed by Hewlett-Packard. The Basic system is not generally acceptable for testing since it is too slow. Pascal is much faster than Basic since it is a pre-compiled language. The procedure/library system also aids in software development, since components can be developed first, then a main program can combine all the components. Generic components can also be used for many different programs.

The HP-6940B multiprogrammer is the reason for selecting Hewlett-Packard as the vendor; The multiprogrammer can read strain gages and thermocouples, which are in the micro-volt range, directly without any pre-amps required. Specifically, the low level A/D card required for reading strain gages direct without amplifiers.

Multiprogrammer has wide range of cards available, enough to imitate large system on smaller scale:

- D/A cards available for controlling test machines, heat chambers, etc.
- High speed A/D card and quad A/D for reading test machines and other devices that have a 0-10 volt range.
- Low level A/D for reading strain gages and thermocouples.

- Relay open cards for controlling test machines, incrementing counters, or triggering other devices.
- Event sense cards for sensing errors, machine status, or switches.

As many of each of the above cards can be used, up to twelve cards total, depending on the application.

There are many significant advantages to the HP-9826 based system:

- Tailorability: Additional cards can be purchased for the multiprogrammer if required for an individual test at a relatively small cost.
- Portability: These systems can be rack mounted, which make them easy to move to any location in the lab or test floor. The computer itself can even be moved to the office for data reduction or report generation.
- Reliability: When all systems are not being used, if one computer goes down, another can quickly replace it.
- Low Cost: Several systems can be purchased for the price of one main-frame system. In the case of the Structural Test Facility, having five systems allow up to four different tests to be run, while one system is utilized for software development, or in an emergency, five tests can be run.
- More Modern: The HP-9826 computer has a time based generator built in, so a sine wave function can be calculated, and a haversine method of loading doesn't have to be used as with the case of the HP-1000.

There are, of course, some disadvantages to the HP-9826 system:

- **The amount of RAM in the HP-9826 limits the size of fatigue spectrum.**
Four memory cards can be used in each machine, and at least two are required to run Pascal and the applications software. Typically, each machine does not require the maximum number of boards to run a test, so they can be swapped around from machine to machine as required. However, the average aircraft spectrum requires maximum memory, but we have not had any spectrum yet so large we couldn't handle it. Also, the maximum number of boards are required to simplify software development.
- **The throughput of the low level A/D card is extremely slow. Each channel is read sequentially, not parallel like on large system; therefore, the more strain gages, the slower it runs. As a rule of thumb, the speed is adequate for tests of five strain gages or less. Any more strain gages than that and there is a risk of losing the data at failure. It is also not recommended on fatigue tests since failure may occur while at peak load, and any gages not read before failure would be lost. Also, reading too many strain gages combined with a fast loading rate causes sine function to become discrete steps, similar to a haversine, which causes chugging. This can be avoided by reading strain gages only at prescribed steps, which again may prevent getting strain data at failure.**
- **In addition to being slow, there is also a limiting number of strain gages that can be read since one card is required for every ten strain gages. A large scale test can have 100 to 200 strain gages; this system should really be**

limited to 10 strain gages, although it can theoretically handle as many as 50.

- The number of load channels that can be read during a test are also limited for the same reason as the strain gage channels; the channels are read sequential, not parallel. A realistic limit is three to five channels of load control to maintain a reasonable throughput with a desirable level of safety.

A.6 Description of the Hybrid HP-9826/HP-1000 system

These disadvantages of the HP-9826 described above, while significant, are not insurmountable. One solution that has been used successfully on several test programs such as the High Strain Wing Repair has been to link the HP-9826 to the HP-1000. The HP-9826 is used to control the test, and the HP-1000 is used for data acquisition. Specifically:

- The HP-9826 is tied to the servacs to control the load and the load feedback to verify that the load is within tolerance. The HP-9826 is used to perform the random spectrum fatigue tests, and to control the static test calibrations.
- The HP-1000 is for data acquisition during the static test calibrations. Load cells, strain gages, and extensimeters are read by the HP-1000 when commanded by the HP-9826. The HP-9826 sends a signal to the HP-1000 at each load increment, and the HP-1000 records the data on 9-track tape. The HP-9826 sends a different signal when a load point is reached. The operator designates the load points, and typically uses 5 or 10% of design limit load as the interval between load points. There are

between 50 to 100 load increments between each load point, depending on the loading rate. The load is held during the load point so that the data can be reviewed, and the loads on each actuator can be manually adjusted if required. Using load points that are 5 to 10% of the design limit load also reduces the amount of data, which simplifies the post-test analysis and plotting, while the load increment data on 9-track tape allows for failure analysis in the event of a catastrophic failure or a non-linearity in the strain or extensimeter data.

The resulting hybrid system gives the most flexibility. The HP-9826 can run the fatigue test unattended, while the HP-1000 is used to perform post-test analysis, software development, or even perform a different test. The drawback to this hybrid system is that there is no strain data recorded during the fatigue test. If a failure occurs during the fatigue test, there will not be any strain data to assist in the post-failure analysis.

APPENDIX B

CSCI-1 HIGH STRAIN WING REPAIR TEST PROGRAM LISTING

APPENDIX B.1

CSC-1.1 Test Executive

```

1:D      0 $UCSD$ $LINES 55$ $DEBUG ON $ $CODE ON $
2:D      0 $REF 75$ $LIST ON $ $TABLES OFF$
3:D      0 $SEARCH '#47:PASC_LIB.', '#3:TEST_PARM',
4:D      0          '#3:TEST_COND', '#3:PUMP_CMDS',
5:D      0          '#3:LOAD_CMDS', '#3:VAR_DEFIN'$
6:S
7:D      0 PROGRAM HSWR;
8:S
9:D      1 {*****}
10:D     1 {***                                     ***}
11:D     1 {*** This program runs a fatigue test for the ***}
12:D     1 {*** High Strain Wing Repair panel in the 300 ***}
13:D     1 {*** kip horizontal actuator fixture.         ***}
14:D     1 {***                                     ***}
15:D     1 {*** Written by R. Dalrymple                   ***}
16:D     1 {***                                     ***}
17:D     1 {*** Revision List:                             ***}
18:D     1 {***                                     ***}
19:D     1 {*** 1.0 12 April 1988                         ***}
20:D     1 {***                                     ***}
21:D     1 {*****}
22:S
23:S
24:S
25:S

```



```

26:D      1 $PAGE$
27:D      1
28:D      1 LABEL 10,15,20,25,30,35,40,45,90;
29:S
30:D      1 IMPORT
31:D      1     GENERAL_0,GENERAL_1,GENERAL_2,GENERAL_3,
32:D      1     CLOCK,IODECLARATIONS,SYSGLOBALS,IOCOMASM,
33:D      1     TEST_PARAMETERS,TEST_CONDITIONS,
34:D      1     PUMP_COMMANDS,LOAD_COMMANDS,VAR_DEFINITIONS;
35:S
36:D      1 CONST
37:D      1     CORR_VAL=1.08; { Correction to positive loads }
38:S
39:D      1 VAR
40:D      1     I,IOR,P,
41:D      1     PASS,FLIGHT,POINT,TOTPASS,
42:D      1     OLDPASS,OLDFLT,OLDPT,
43:D      1     LBSMAX, { Maximum actuator load }
44:D      1     DLL, { Design limit load }
45:D      1     LDRATE, { Loading rate }
46:D      1     LD_TOL, { Load tolerance }
47:D      1     UPRBND, { Upper bound that triggers dump }
48:D      1     LWRBND, { Lower bound that triggers dump }
49:D      1     CAL_DELAY, { Used in delaying screen update }
50:D      1     SPEC_VAL, { Value read from spectrum }
51:D      1     FDBACK, { Load reading from load cell }
52:D      1     { Ramping goes from STRTPC to NEXTPC in %DLL }
53:D      1     { Ramping goes from LASTLD to LOADPT in lbs. }
54:D      1     LOADPT, { Loadpt=Nextpt converted to pounds }
55:D      -84 1     LASTLD:INTEGER; { Previous Loadpt }
56:D      -84 1     FS, { Load cell full scale voltage }
57:D      -84 1     NEXTPC, { Corrected load point in %DLL }
58:D      -84 1     STRTPC, { Previous Nextpc }
59:D      -84 1     DEL_PC, { Difference of Nextpc and Strtpc }
60:D      -84 1     PC_DLL, { Load output to actuator in %DLL }
61:D      -84 1     PC_DIF, { Percent difference in load at Hold }
62:D      -84 1     OMEGA, { Load rate converted to frequency }
63:D      -84 1     TO,T, { Starting/current time during Ramp }
64:D      -164 1     SIN_TERM:REAL; { Value of Sine during Ramp }
65:D      -165 1     REPLY :CHAR; { Operators reply to questions }
66:D      -168 1     TIME :TIMEREC; { System time }
67:D      -250 1     TITLE :S80; { Title for CRT }
68:D      -250 1     CNFG_CHNG,{ Flag for configuration change }
69:D      -250 1     FATIGUE, { Flag for fatigue test }
70:D      -250 1     REPLY_OK, { Used on check operator on respnse }
71:D      -250 1     OP_STOP, { Operator set switch to stop test }
72:D      -255 1     REF_SCR:BOOLEAN; { Flag to refresh screen }
73:D      -918 1     DATA_FILE:TEXT; { Spectrum Data File }
74:S
75:S

```

```

76:D -91B 1 $PAGE$
77:S
78:C 1 BEGIN
79:S
500:C 1 $Linenum 500$
501:S
502*C 1 FATIGUE:=TRUE;
503*C 1 CNFG_CHNG:=FALSE;
504*C 1 OUTPUT_LOAD(0,100000);
505:C 1 { Open data file }
506:C 1 $IOCHECK OFF$
507*C 1 RESET(DATA_FILE,'#48:WS125.TEXT');
508:C 1 $IOCHECK ON $
509*C 1 IOR:=IORESULT;
510*C 1 IF IOR<>0 THEN
511:C 2 BEGIN
512*C 2 WRITE (FF,LF,LF,' Can''t open memory ');
513*C 2 WRITELN('volume, IOresult=',IOR:1,LF);
514*C 2 WRITE (' Hit any key to continue....');
515*C 2 READ (REPLY);
516:C 2 END;
517:S
1000:C 1 $Linenum 1000$
1001:S
1002:C 1 10:REPEAT
1003:C 2 { Write menu }
1004*C 2 IF FATIGUE THEN
1005*C 3 TITLE:=FATIGUETITLE
1006:C 3 ELSE
1007*C 3 TITLE:=STATIC_TITLE;
1008:C 2
1009*C 2 WRITELN(FF,TITLE,LF,LF);
1010:S
1011*C 2 IF FATIGUE THEN
1012*C 3 {0} WRITELN(STRO,'Static Test',LF)
1013:C 3 ELSE
1014*C 3 WRITELN(STRO,'Fatigue Test',LF);
1015:S
1016*C 2 {1} WRITELN(STR1,LF);
1017*C 2 {2} WRITELN(STR2,LF);
1018*C 2 {3} WRITELN(STR3,LF);
1019*C 2 {4} WRITELN(STR4,LF);
1020:S
1021*C 2 {5} IF DUMPED THEN
1022*C 3 WRITELN(STR5,LF)
1023:C 3 ELSE
1024*C 3 WRITELN(' 5.',LF);
1025:S
1026*C 2 {6} WRITELN(STR6,LF);
1027*C 2 {7} WRITELN(STR7,LF);
1028:S
1029*C 2 {8} IF FATIGUE THEN
1030:C 3 BEGIN
1031*C 3 WRITE (STR8,'Change number of passes');
1032*C 3 IF TOTPASS>0 THEN

```

```

1033*C      4      WRITE(' , currently ',TOTPASS:1);
1034*C      3      WRITELN(LF);
1035:C      3      END
1036:C      3      ELSE
1037*C      3      WRITELN(STRB,LF);
1038:S
1039*C      2      IF NOT(DUMPED) THEN
1040*C      3 {9}    WRITELN(STR9);
1041:S
1042:C      2      { Read request }
1043*C      2      WRITE (LF,'Request?');
1044*C      2      READ  (REPLY);
1045*C      2      WRITELN(FF);
1046:S
1047*C      2 {0}    IF REPLY='0' THEN
1048*C      3      FATIGUE:=NOT(FATIGUE);
1049:S
1050*C      2 {1}    IF REPLY='1' THEN
1051*C      3      READ_CONFIG (FATIGUE,FS,LBSMAX,DLL,LDRATE,
1052:C      3      LD_TOL,CAL_DELAY,PASS,FLIGHT,POINT,TOTPASS);
1053:S
1054*C      2 {2}    IF REPLY='2' THEN
1055:C      3      BEGIN
1056*C      3      CHNG_SYST_PARAM (FS,LBSMAX,DLL);
1057*C      3      CNFG_CHNG:=TRUE;
1058:C      3      END;
1059:S
1060*C      2 {3}    IF REPLY='3' THEN
1061:C      3      BEGIN
1062*C      3      CHNG_TEST_PARAM (LDRATE,LD_TOL);
1063*C      3      CNFG_CHNG:=TRUE;
1064:C      3      END;
1065:S
1066*C      2 {4}    IF REPLY='4' THEN
1067:C      3      BEGIN
1068*C      3      WRITE_CONFIG (FATIGUE,FS,LBSMAX,DLL,LDRATE,
1069:C      3      LD_TOL,CAL_DELAY,PASS,FLIGHT,POINT,TOTPASS);
1070*C      3      CNFG_CHNG:=FALSE;
1071:C      3      END;
1072:S
1073*C      2 {5}    IF REPLY='5' THEN { Power_Up }
1074*C      3      POWER_UP;
1075:S
1076*C      2 {6}    IF REPLY='6' THEN { Cal load reading }
1077*C      3      CAL_LOAD(FS,CAL_DELAY);
1078:S
1079*C      2 {7}    IF REPLY='7' THEN
1080*C      3      IF CNFG_CHNG THEN
1081:C      4      BEGIN
1082*C      4      IF PROGRAM_END(FATIGUE) THEN
1083*C      5      GOTO 90
1084:C      5      END
1085:C      4      ELSE
1086*C      4      GOTO 90;
1087:S

```

```

1088*C      2 {8} IF (REPLY='B') AND FATIGUE THEN
1089:C      3     BEGIN
1090*C      3         NUMBER_PASSES(TOTPASS);
1091*C      3         CNFG_CHNG:=TRUE;
1092:C      3     END;
1093:S
1094*C      2 {9} UNTIL (REPLY='9') AND NOT(DUMPED);
1095:C      1
1096:S
1097:S
1098:C      1     { ***** Initialize Values ***** }
1099:S
1100*C      1     NEXTPC:=0;  STRTPC:=0;
1101*C      1     LOADPT:=0;  LASTLD:=0;
1102*C      1     UPRBND:= LD_TOL;
1103*C      1     LWRBND:=-LD_TOL;
1104:S
1105:S
1106:S
1107:C      1     { ***** Compute load factors for screen ***** }
1108:S
1500:C      1 $Linenum 1500$
1501:S
1502*C      1 15: STRTPC:=0;
1503*C      1     LASTLD:=0;
1504:S
1505*C      1     OP_STOP:=FALSE;
1506*C      1     REF_SCR:=FALSE;
1507:S
1508:C      1     { Static or Fatigue? }
1509*C      1     IF NOT(FATIGUE) THEN
1510:C      2     BEGIN { Static Test }
1511*C      2         NEXTPC:=0;
1512*C      2         LOADPT:=0;
1513*C      2         GOTO 40;
1514:C      2     END;
1515:S
1516:C      1     { Fatigue Test }
1517:S
1518*C      1     OLDPASS:=PASS;
1519*C      1     OLDFLT:=FLIGHT;
1520*C      1     OLDPT:=POINT;
1521*C      1     RESTART (PASS,FLIGHT,POINT,TOTPASS);
1522:C      1     { Reset counters }
1523*C      1     IF NOT((PASS=OLDPASS) AND (FLIGHT=OLDFLT)
1524:C      2             AND (POINT=OLDPT)) THEN
1525:C      2     BEGIN
1526*C      2         WRITE (FF,LF,LF,
1527:C      2             '      Resetting counters, please wait....');
1528*C      2         RESET_COUNTERS(PASS,FLIGHT,POINT+1);
1529:C      2     END;
1530:C      1     { Reset file pointer }
1531*C      1     P:=(FLIGHT-1)*108+POINT-1;
1532*C      1     IF NOT((FLIGHT=OLDFLT) AND (POINT=OLDPT))
1533:C      2         AND (P<>0) THEN

```

```

1534:C      2      BEGIN
1535*C      2          WRITE (FF,LF,LF,'    Setting file pointer,',
1536:C      2              ' please wait....');
1537*C      2          RESET(DATA_FILE,'#48:WS125.TEXT');
1538*C      2          FOR I:=1 TO P DO
1539*C      3              READ(DATA_FILE,SPEC_VAL);
1540:C      2      END;
1541:S
1542:S
1543:C      1          { Find load values }
1544*C      1      READ(DATA_FILE,SPEC_VAL);
1545*C      1      NEXTPC:=SPEC_VAL/10;
1546:C      1          { Grumman/Boeing Vertol Cor- }
1547:C      1          { rection for positive loads }
1548*C      1      IF NEXTPC>0 THEN
1549*C      2          NEXTPC:=NEXTPC*CORR_VAL;
1550*C      1      LOADPT:=ROUND(NEXTPC*DLL/100);
1551:S
1552*C      1      CNFG_CHNG:=TRUE;
1553:C      1
1554:C      1          { Hold at starting point as a final check }
1555*C      1      WRITELN(FF,TITLE,LF,LF);
1556*C      1      WRITELN(TB,'    Pass Number =',PASS:8,LF);
1557*C      1      WRITELN(TB,'    Flight Number =',FLIGHT:8,LF);
1558*C      1      WRITELN(TB,'    Point Number =',POINT:8,LF);
1559*C      1      WRITELN(TB,'    Spectrum Value =',
1560:C      1          (SPEC_VAL/10):8:1,' %DLL',LF);
1561*C      1      WRITELN(TB,'    Corrected Value =',NEXTPC:8:1,' %DLL',LF);
1562*C      1      WRITELN(TB,'    Desired Load =',LOADPT:8,' lbs',TD);
1563*C      1      WRITELN(TB,'    Hit any key to start test, or',LF);
1564*C      1      WRITELN(TB,'    M to return to menu....');
1565*C      1      WRITE (TU,US,US,TB,'    Feedback =',LOAD(FS):8,
1566:C      1          ' lbs',BS,BS,BS,BS);
1567:S
1568:C      1      REPEAT
1569*C      2          SYSTIME(TIME);
1570*C      2          WITH TIME DO
1571*C      3              TO:=HOUR*3600+MINUTE*60+CENTISECOND/100;
1572:C      2          REPEAT
1573*C      3              SYSTIME(TIME);
1574*C      3              WITH TIME DO
1575*C      4                  T:=HOUR*3600+MINUTE*60+CENTISECOND/100;
1576*C      3          UNTIL (T-TO)>0.075; { 75 msec delay }
1577*C      2          FDBACK:=LOAD(FS);
1578*C      2          WRITE (BT,FDBACK:8);
1579:S
1580:C      2          { Check for overload }
1581*C      2          IF (FDBACK>UPRBND) OR
1582:C      3              (FDBACK<LWRBND) THEN
1583:C      3              BEGIN
1584*C      3                  DUMP;
1585:C      3                  { Clear unit if key was hit }
1586*C      3                  IF NOT(UNITBUSY(KEYBRD)) THEN
1587:C      4                      BEGIN
1588*C      4                          READ(REPLY); { Character read is not used }

```

```

1589*C      4      WRITE (BS,SP);
1590:C      4      END;
1591*C      3      WRITELN(BL,LF,LF);
1592*C      3      IF NEXTPC<>0 THEN
1593*C      4      WRITE (TD);
1594*C      3      IF FDBACK>UPRBND THEN
1595:C      4      BEGIN
1596*C      4      WRITELN(' Test stopped on Positive overload');
1597*C      4      WRITELN(' Upper bound = ',UPRBND:1,'lbs',TB,TB);
1598:C      4      END
1599:C      4      ELSE
1600:C      4      BEGIN
1601*C      4      WRITELN(' Test stopped on Negative overload');
1602*C      4      WRITELN(' Lower bound = ',LWRBND:1,'lbs',TB,TB);
1603:C      4      END;
1604*C      3      WRITE (LF,TB,TB,TB,US,CR);
1605*C      3      WRITE (' Hit any key to continue....',TB,BT,BS,BS);
1606*C      3      READ (REPLY);
1607*C      3      GOTO 10; { Go to Menu }
1608:C      3      END;
1609:S
1610*C      2      UNTIL NOT(UNITBUSY(KEYBRD));
1611:S
1612*C      1      READ (REPLY);
1613*C      1      IF (REPLY='M') OR (REPLY='m') THEN
1614*C      2      GOTO 10; { Go to Menu }
1615:S
1616*C      1      IF NEXTPC=STRTPC THEN { Increment counters }
1617:C      2      BEGIN
1618*C      2      REF_SCR:=TRUE;
1619*C      2      GOTO 35;
1620:C      2      END;
1621:C      1
1622:S
1623:S
1624:C      1      ( '***** Refresh Screen ***** )
1625:S
2000:C      1 $Linenum 2000$
2001:S
2002*C      1 20:WRITELN(FF,TITLE,LF,LF);
2003*C      1      IF FATIGUE AND NOT(OP_STOP) THEN
2004:C      2      BEGIN
2005*C      2      WRITELN(TB,' Pass Number =',PASS:8,LF);
2006*C      2      WRITELN(TB,' Flight Number =',FLIGHT:8,LF);
2007*C      2      WRITELN(TB,' Point Number =',POINT:8,LF);
2008*C      2      WRITELN(TB,' Spectrum Value =',
2009:C      2      (SPEC_VAL/10):8:1,' %DLL',LF);
2010*C      2      WRITELN(TB,'Corrected Value =',
2011:C      2      NEXTPC:8:1,' %DLL',LF);
2012:C      2      END
2013:C      2      ELSE
2014:C      2      BEGIN
2015*C      2      WRITELN(TB,' Ramping to ',NEXTPC:5:1,
2016:C      2      '% of DLL',LF);
2017*C      2      WRITELN(TB,TB,' ',(2*NEXTPC/3):5:1,

```

```

2018:C      2      '% of TUL',LF,LF);
2019:C      2      END;
2020:S
2021*C      1      WRITELN(TB,'  Desired Load =',LOADPT:8,' lbs',TD);
2022*C      1      WRITELN(TB,' Hit any key to halt test....');
2023*C      1      WRITE  (TU,TB,'          Feedback =',LOAD(FS):8,
2024:C      1      ' lbs',BS,BS,BS,BS);
2025:S
2026*C      1      REF_SCR:=FALSE;
2027:S
2028:S
2029:S
2030:C      1      { ***** Compute Load Factors ***** }
2031:S
2500:C      1      $Linenum 2500$
2501:S
2502*C      1      25:DEL_PC:=NEXTPC-STRTPC;
2503*C      1      OMEGA:=ABS(LDRATE/(2*DEL_PC/100*DLL));
2504*C      1      IF (LOADPT-LASTLD)>0 THEN
2505:C      2      BEGIN
2506*C      2      UPRBND:=LOADPT+LD_TOL;
2507*C      2      LWRBND:=LASTLD-LD_TOL;
2508:C      2      END
2509:C      2      ELSE
2510:C      2      BEGIN
2511*C      2      UPRBND:=LASTLD+LD_TOL;
2512*C      2      LWRBND:=LOADPT-LD_TOL;
2513:C      2      END;
2514:S
2515:C      1      { Find starting time }
2516*C      1      SYSTIME(TIME);
2517*C      1      WITH TIME DO
2518:C      2      TO:=HOUR*3600+MINUTE*60+CENTISECOND/100;
2519:S
2520:S
2521:S
2522:C      1      { ***** Ramp ***** }
2523:S
3000:C      1      $Linenum 3000$
3001:S
3002:C      1      30: { Determine value of next load step }
3003*C      1      SYSTIME(TIME);
3004*C      1      WITH TIME DO
3005*C      2      T:=HOUR*3600+MINUTE*60+CENTISECOND/100-T0;
3006*C      1      IF (OMEGA*T)<0.5 THEN
3007*C      2      SIN_TERM:=0.5*(SIN(PI*(2*OMEGA*T-0.5))+1)
3008:C      2      ELSE
3009*C      2      SIN_TERM:=1.0;
3010:S
3011:C      1      { Step up loads, read/display loads }
3012*C      1      PC_DLL:=DEL_PC*SIN_TERM+STRTPC;
3013*C      1      OUTPUT_LOAD(PC_DLL*DLL/100,LBSMAX);
3014*C      1      SCAN_1000 (FALSE);
3015*C      1      FDBACK:=LOAD(FS);
3016*C      1      WRITE  (BT,FDBACK:8);

```

```

3017:S
3018:C      1      { Check for overload }
3019*C      1      IF (FDBACK>UPRBND) OR
3020:C      2      (FDBACK<LWRBND) THEN
3021:C      2      BEGIN
3022*C      2      DUMP;
3023:C      2      { Clear unit if key was hit }
3024*C      2      IF NOT(UNITBUSY(KEYBRD)) THEN
3025:C      3      BEGIN
3026*C      3      READ(REPLY);      { Character read is not used }
3027*C      3      WRITE (BS,SP);
3028:C      3      END;
3029*C      2      WRITELN(BL,LF,LF);
3030*C      2      IF FDBACK>UPRBND THEN
3031:C      3      BEGIN
3032*C      3      WRITELN(' Test stopped on Positive overload');
3033*C      3      WRITELN(' Upper bound = ',UPRBND:1,'lbs');
3034:C      3      END
3035:C      3      ELSE
3036:C      3      BEGIN
3037*C      3      WRITELN(' Test stopped on Negative overload');
3038*C      3      WRITELN(' Lower bound = ',LWRBND:1,'lbs');
3039:C      3      END;
3040*C      2      WRITE (' Hit any key to continue....');
3041*C      2      READ (REPLY);
3042*C      2      GOTO 10; { Go to Menu }
3043:C      2      END;
3044:S
3045:C      1      { Check for external dump }
3046*C      1      IF DUMPED=TRUE THEN {Machine is in dump}
3047:C      2      BEGIN
3048*C      2      IF NOT(UNITBUSY(KEYBRD)) THEN { Clear unit }
3049*C      3      READ(REPLY);      { Character read is not used }
3050*C      2      WRITELN(BL,CR,LF,LF,TB,'External dump at ',
3051:C      2      PC_DLL:2:2,'% of DLL');
3052*C      2      WRITELN(TB,' Hit any key to zero command signal');
3053*C      2      WRITE (TB,' and continue....');
3054*C      2      READ (REPLY);
3055*C      2      DUMP;
3056*C      2      GOTO 10; { Go to Menu }
3057:C      2      END;
3058:S
3059:C      1      { Operator want test halted? }
3060*C      1      IF NOT(UNITBUSY(KEYBRD)) THEN
3061:C      2      BEGIN      { Any key hit will halt test }
3062*C      2      READ(REPLY);      { Character read is not used }
3063*C      2      NEXTPC:=PC_DLL;
3064*C      2      LOADPT:=ROUND(DLL*PC_DLL/100);
3065*C      2      GOTO 40;      { Go to Hold }
3066:C      2      END;
3067:S
3068:C      1      { Reached load point? }
3069*C      1      IF SIN_TERM<1.0 THEN
3070*C      2      GOTO 30; { Go to Ramp }
3071:S

```



```

3072*C      1      IF OP_STOP THEN
3073:C      2      { Dump pump if switch is set }
3074*C      2      IF SWITCH_SET(2) THEN
3075:C      3      BEGIN
3076*C      3      DUMP;
3077*C      3      Writeln(FF,TD,TB,'Test stopped and pump dumped');
3078*C      3      Writeln(TB,'per operator request. ');
3079*C      3      WRITE (TD,TB,' Hit any key to continue.... ');
3080*C      3      READ (REPLY);
3081*C      3      GOTO 10; { Go to Menu }
3082:C      3      END;
3083:S
3084*C      1      IF NOT(FATIGUE) OR OP_STOP THEN
3085*C      2      GOTO 40; { Hold if Static test or switch set }
3086:S
3087*C      1      IF NEXTPC>STRIPC THEN
3088:C      2      BEGIN
3089*C      2      IF SWITCH_SET(5) THEN
3090*C      3      GOTO 40; { Hold at pass }
3091:C      2      END;
3092:C      2      ELSE
3093*C      2      IF SWITCH_SET(4) THEN
3094*C      3      GOTO 40; { Hold at valley }
3095:S
3096:S
3097:S
3098:C      1      { **** Increment Counters **** }
3099:C
3100:C      1      { Minimum 2000 }
3101:S
3102:C      1      3511:=1+1;
3103*C      1      POINT:=POINT+1;
3104*C      1      WRITE (TU,TU);
3105*C      1      IF POINT=109 THEN { New Flight }
3106:C      2      BEGIN
3107*C      2      POINT:=1;
3108*C      2      FLIGHT:=FLIGHT+1;
3109*C      2      WRITE (US,US);
3110*C      2      OP_STOP:=FALSE;
3111*C      2      IF FLIGHT=356 THEN { New Pass }
3112:C      3      BEGIN
3113*C      3      FLIGHT:=1;
3114*C      3      PASS:=PASS+1;
3115*C      3      IF SWITCH_SET(0) THEN { Hold at end of pass }
3116*C      4      OP_STOP:=TRUE;
3117*C      3      WRITE (BT,US,US,PASS:8,LF,LF);
3118*C      3      INCR_COUNTER(3);
3119*C      3      RESET(DATA_FILE,'#48:WS125.TEXT');
3120:C      3      END;
3121*C      2      IF SWITCH_SET(1) THEN { Hold at end of flight }
3122*C      3      OP_STOP:=TRUE;
3123*C      2      WRITE (BT,FLIGHT:8,LF,LF);
3124*C      2      INCR_COUNTER(2);
3125:C      2      END;
3126:S

```

```

3527*C      1      WRITE (BT,POINT:8);
3528*C      1      INCR_COUNTER(1);
3529:S
3530*C      1      STRTPC:=NEXTPC;
3531*C      1      LASTLD:=LOADPT;
3532:S
3533:C      1      { If Switch 0 or 1 is set, automatically unload }
3534*C      1      IF OP_STOP THEN
3535:C      2      BEGIN
3536*C      2          NEXTPC:=0;
3537*C      2          LOADPT:=0;
3538*C      2          GOTO 20;
3539:C      2      END;
3540:S
3541:C      1      { Find load values }
3542*C      1      READ(DATA_FILE,SPEC_VAL);
3543*C      1      NEXTPC:=SPEC_VAL/10;
3544:C      1      { Grumman/Boeing Vertol Cor- }
3545:C      1      { rection for positive loads }
3546*C      1      IF NEXTPC>0 THEN
3547*C      2          NEXTPC:=NEXTPC*CORR_VAL;
3548*C      1      LOADPT:=ROUND(NEXTPC*DLL/100);
3549:S
3550*C      1      IF NEXTPC=STRTPC THEN
3551*C      2          GOTO 35; { Increment counters if same value }
3552:S
3553*C      1      IF REF_SCR THEN
3554*C      2          GOTO 20; { Refresh Screen }
3555:S
3556*C      1      WRITE (BT,POINT:8,LF,LF);
3557*C      1      WRITE (BT,(SPEC_VAL/10):8:1,LF,LF);
3558*C      1      WRITE (BT,NEXTPC:8:1,LF,LF);
3559*C      1      WRITE (BT,LOADPT:8,LF,LF);
3560*C      1      WRITE (BT,FDBACK:8);
3561:S
3562*C      1      GOTO 25;
3563:S
3564:S
3565:S
3566:C      1      { ***** Hold ***** }
3567:S
4000:C      1 $Linenum 4000$
4001:S
4002*C      1 40:STRTPC:=NEXTPC;
4003*C      1      LASTLD:=LOADPT;
4004*C      1      UPRBND:=LOADPT+LD_TOL;
4005*C      1      LWRBND:=LASTLD-LD_TOL;
4006:S
4007*C      1      WRITELN(FF,TITLE,LF,LF);
4008*C      1      WRITELN(TB,' Holding at ',NEXTPC:5:1,
4009:C      1          '% of DLL',LF);
4010*C      1      WRITELN(TB,TB,' ',(2*NEXTPC/3):5:1,
4011:C      1          '% of TUL',LF,LF);
4012*C      1      WRITELN(TB,' Desired Load =',LOADPT:8,
4013:C      1          ' lbs',TD);

```

```

4014*C      1      IF NEXTPC<>0 THEN WRITELN(LF,LF,LF);
4015*C      1      WRITELN(TB,' Hit:');
4016:S
4017*C      1      IF FATIGUE THEN
4018*C      2          WRITELN(TB,'      C to Continue test were stopped')
4019:C      2      ELSE
4020:C      2      BEGIN
4021*C      2          WRITELN(TB,'      S to trigger data Scan');
4022*C      2          WRITELN(TB,'      L for next Load point');
4023:C      2      END;
4024:S
4025*C      1      IF NEXTPC=0 THEN
4026*C      2          WRITELN(TB,'      M to return to the Menu ',US)
4027:C      2      ELSE
4028*C      2          WRITELN(TB,'      U to Unload      ',TB,TU,US);
4029:S
4030*C      1      WRITE (TU,US);
4031*C      1      IF NOT(FATIGUE) THEN WRITE(US);
4032:S
4033*C      1      FDBACK:=LOAD(FS);
4034*C      1      WRITE (TB,'      Feedback =',FDBACK:B,
4035:C      1          ' lbs',BS,BS,BS,BS);
4036*C      1      IF NEXTPC<>0 THEN
4037:C      2      BEGIN
4038*C      2          WRITELN(LF);
4039*C      2          WRITELN(T5,'Absolute Difference =',
4040:C      2              (FDBACK-LOADPT):B,' lbs',LF);
4041*C      2          PC_DIF:=ABS(1-FDBACK/LOADPT)*100;
4042*C      2          IF PC_DIF<0.1 THEN PC_DIF:=0;
4043*C      2          WRITE (T5,' Percent Difference =',
4044:C      2              PC_DIF:B:1,' %',BS,BS,TU);
4045:C      2      END;
4046:S
4500:C      1 $Linenum 4500$
4501:S
4502:C      1 45: { Inner loop repeats until operator }
4503:C      1 { responds, and Outer loop repeats }
4504:C      1 { until operator response is ok }
4505:C      1 REPEAT { Outer loop }
4506:C      2 REPEAT { Inner Loop }
4507*C      3 FDBACK:=LOAD(FS);
4508*C      3 WRITE (BT,FDBACK:B);
4509*C      3 IF NEXTPC<>0 THEN
4510:C      4 BEGIN
4511*C      4 WRITE (LF,LF,BT,(FDBACK-LOADPT):B,LF,LF);
4512*C      4 PC_DIF:=ABS(1-FDBACK/LOADPT)*100;
4513*C      4 IF PC_DIF<0.1 THEN PC_DIF:=0;
4514*C      4 WRITE (BT,PC_DIF:B:1,TU);
4515:C      4 END;
4516:S
4517*C      3 SYSTIME(TIME);
4518*C      3 WITH TIME DO
4519*C      4 TO:=HOUR*3600+MINUTE*60+CENTISECOND/100;
4520:C      3 REPEAT
4521*C      4 SYSTIME(TIME);

```

```

4522*C      4      WITH TIME DO
4523*C      5      T:=HOUR*3600+MINUTE*60+CENTISECOND/100;
4524*C      4      UNTIL (T-T0)>0.075; { 75 msec delay }
4525:S
4526:C      3      { Check for overload }
4527*C      3      IF (FDBACK>UPRBND) OR
4528:C      4      (FDBACK<LWRBND) THEN
4529:C      4      BEGIN
4530*C      4      DUMP;
4531:C      4      { Clear unit if key was hit }
4532*C      4      IF NOT(UNITBUSY(KEYBRD)) THEN
4533:C      5      BEGIN
4534*C      5      READ(REPLY); { Character read is not used }
4535*C      5      WRITE (BS,SP);
4536:C      5      END;
4537*C      4      WRITELN(BL,LF,LF);
4538*C      4      IF NEXTPC<>0 THEN
4539*C      5      WRITE (TD);
4540*C      4      IF FDBACK>UPRBND THEN
4541:C      5      BEGIN
4542*C      5      WRITELN(
4543:C      5      ' Test stopped on Positive overload');
4544*C      5      WRITELN(
4545:C      5      ' Upper bound = ',UPRBND:1,'lbs',TB,TB);
4546:C      5      END
4547:C      5      ELSE
4548:C      5      BEGIN
4549*C      5      WRITELN(
4550:C      5      ' Test stopped on Negative overload');
4551*C      5      WRITELN(
4552:C      5      ' Lower bound = ',LWRBND:1,'lbs',TB,TB);
4553:C      5      END;
4554*C      4      WRITE (LF,TB,TB,TB,US,CR);
4555*C      4      WRITE (' Hit any key to continue....',
4556:C      4      TB,BT,BS,BS);
4557*C      4      READ (REPLY);
4558*C      4      GOTO 10; { Go to Menu }
4559:C      4      END;
4560:S
4561*C      3      UNTIL NOT(UNITBUSY(KEYBRD)); { Inner loop }
4562:S
4563*C      2      READ(REPLY);
4564*C      2      WRITE(BS,SP,BS);
4565:S
4566*C      2      IF NEXTPC=0 THEN
4567*C      3      REPLY_OK:=((REPLY='M') OR (REPLY='m'))
4568:C      3      ELSE
4569*C      3      REPLY_OK:=((REPLY='U') OR (REPLY='u'));
4570:S
4571*C      2      IF FATIGUE THEN
4572:C      3      BEGIN
4573*C      3      IF (REPLY='C') OR (REPLY='c') THEN
4574*C      4      REPLY_OK:=TRUE;
4575:C      3      END
4576:C      3      ELSE

```

```

4577*C      3      IF (REPLY='S') OR (REPLY='s') OR
4578:C      4      (REPLY='L') OR (REPLY='l') THEN
4579*C      4      REPLY_OK:=TRUE;
4580:S
4581:C      2      { Outer loop }
4582*C      2      UNTIL REPLY_OK;
4583:S
4584*C      1      IF (REPLY='M') OR (REPLY='m') THEN
4585*C      2      GOTO 10; { Go to Menu }
4586:S
4587*C      1      IF (REPLY='S') OR (REPLY='s') THEN
4588:C      2      BEGIN
4589*C      2      SCAN_1000 (TRUE);
4590*C      2      GOTO 45; { Continue hold }
4591:C      2      END;
4592:S
4593*C      1      WRITELN(LF,LF);
4594*C      1      IF NEXTPC<>0 THEN WRITELN(LF,LF,LF);
4595:S
4596*C      1      IF (REPLY='C') OR (REPLY='c') THEN
4597:C      2      BEGIN
4598*C      2      OP_STOP:=FALSE;
4599:C      2      { Find load values }
4600*C      2      NEXTPC:=SPEC_VAL/10;
4601:C      2      { Grumman/Boeing Vertol Cor- }
4602:C      2      { rection for positive loads }
4603*C      2      IF NEXTPC>0 THEN
4604*C      3      NEXTPC:=NEXTPC*CORR_VAL;
4605*C      2      IF NEXTPC=STRTPC THEN { Increment counters }
4606:C      3      BEGIN
4607*C      3      REF_SCR:=TRUE;
4608*C      3      GOTO 35;
4609:C      3      END;
4610*C      2      LOADPT:=ROUND(NEXTPC*DLL/100);
4611*C      2      GOTO 20; { Go to Refresh Screen }
4612:C      2      END;
4613:S
4614*C      1      IF (REPLY='U') OR (REPLY='u') THEN
4615:C      2      BEGIN
4616*C      2      NEXTPC:=0;
4617*C      2      LOADPT:=0;
4618*C      2      OP_STOP:=TRUE;
4619*C      2      GOTO 20; { Go to Refresh Screen }
4620:C      2      END;
4621:S
4622*C      1      IF (REPLY='L') OR (REPLY='l') THEN
4623:C      2      BEGIN
4624*C      2      WRITELN(TB,TB,TB,TB);
4625*C      2      WRITELN(LF,TB,TB,TB,TB);
4626*C      2      WRITELN(TB,TB,TB,TB);
4627*C      2      WRITELN(TB,TB,TB,TU);
4628*C      2      WRITE (BL,CR,'What is new load point ',
4629:C      2      'in percent of DLL? ');
4630:S
4631:C      2      REPEAT

```

Pascal [Rev 2.0 10/19/82] HSWR.TEXT

13-Apr-88 14:38:19 Page 15

```

4632:C      3      $IOCHECK OFF$
4633*C      3      READLN(NEXTPC);
4634:C      3      $IOCHECK ON $
4635*C      3      IOR:=IORESULT;
4636*C      3      IF IOR<>0 THEN
4637*C      4          WRITE (CR,LF,'Try again, new load point? ')
4638:C      4      ELSE
4639*C      4          IF ABS(DLL*NEXTPC/100)>LBSMAX THEN
4640:C      5              BEGIN
4641*C      5                  WRITELN(CR,LF,T5,'Maximum load is +/-',
4642:C      5                      LBSMAX:1,'lbs (' ,ROUND(LBSMAX/DLL*100):1,
4643:C      5                      '% DLL)',LF);
4644*C      5                  WRITE (T5,'Try again, new load point? ');
4645*C      5                  IOR:=1;
4646:C      5                  END;
4647*C      3      UNTIL IOR=0;
4648:S
4649*C      2      IF NEXTPC=STRTPC THEN
4650:C      3      BEGIN
4651*C      3          WRITE (BL);
4652*C      3          GOTO 40; ( Go to Hold )
4653:C      3      END;
4654:S
4655*C      2      LOADPT:=ROUND(NEXTPC*DLL/100);
4656:S
4657*C      2      GOTO 20; ( Go to Refresh Screen )
4658:C      2      END;
4659:S
9000:C      1 $Linenum 9000$
9001:S
9002*C      1 90:WRITELN(TD);
9003*C      1      WRITELN(TB,'Program Ended....');
9004*C      1 END.
9005:S
9006:S
9007:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX B.2

CSC-1.2 Variable Definitions

```

1:D      0 $UCSD$ $LINES 55$
2:D      0 MODULE VAR_DEFINITIONS;
3:S
4:D      1 { Converted for use on the High }
5:D      1 { Strain Wing Repair Tests }
6:S
7:D      1 { Revision dated 20 November 1987 }
8:S
9:S
10:D     1 EXPORT
11:S
12:D     1 TYPE
13:D     1   MONTHS=ARRAY[1..12] OF STRING[9];
14:D     1   LD_TYP=ARRAY[1..750] OF INTEGER;
15:D     1   S20=STRING[20];
16:D     1   S80=STRING[80];
17:S
18:D     1 CONST
19:D     1   KEYBRD=2; { ISC for the Keyboard }
20:D     1   PRINTER=701; { Device selector for Printer }
21:D     1   BL=CHR( 7); { ASCII character for bell }
22:D     1   BS=CHR( 8); { ASCII character for back space}
23:D     1   LF=CHR(10); { ASCII character for line feed }
24:D     1   FF=CHR(12); { ASCII character for form feed }
25:D     1   CR=CHR(13); { ASCII char for carriage return }
26:D     1   US=CHR(31); { ASCII character for up space }
27:D     1   SP=CHR(32); { ASCII character for space }
28:D     1   TB=' ' ; { Tab }
29:D     1   TS=' ' ; {5 space tab}
30:D     1   BT=#00B#00B#00B#00B#00B#00B#00B#00B; {Back tab }
31:D     1   TD=#010#010#010#010; {Tab down}
32:D     1   TU=#031#031#031#031; {Tab up }
33:D     1   SCR_RESET=#00B#00B#031#031#031#031;
34:S
35:D     1   THISMONTH=MONTHS['January','February','March',
36:D     1     'April','May','June','July','August',
37:D     1     'September','October','November','December'];
38:S
39:D     1   FATIGUETITLE=' HIGH STRAIN WING REPAIR FATIGUE TEST';
40:D     1   STATIC_TITLE=' HIGH STRAIN WING REPAIR STATIC TEST';
41:D     1   STR0=' 0. Switch to ';
42:D     1   STR1=' 1. Read configuration files';
43:D     1   STR2=' 2. Change System parameters';
44:D     1   STR3=' 3. Change Test parameters';
45:D     1   STR4=' 4. Write configuration files';
46:D     1   STR5=' 5. Power up the pump';
47:D     1   STR6=' 6. Calibrate load reading';
48:D     1   STR7=' 7. End program';
49:D     1   STR8=' 8. ';
50:D     1   STR9=' 9. Begin test';
51:S
52:D     1   PI=3.14159;
53:D     1   DELTAT=0.05; {Time increment for cal routines}
54:S

```



```
55:D      1  $Page$
56:D      1
57:D      1      Slot0 = 0;      { 400 }
58:D      1      Slot1 = 4096;   { 401 }
59:D      1      Slot2 = 8192;   { 402 }
60:D      1      Slot3 = 12288;  { 403 }
61:D      1      Slot4 = 16384;  { 404 }
62:D      1      Slot5 = 20480;  { 405 }
63:D      1      Slot6 = 24576;  { 406 }
64:D      1      Slot7 = 28672;  { 407 }
65:D      1      Slot8 = -32768; { 408 }
66:D      1      Slot9 = -28672; { 409 }
67:D      1      Slot10=-24576; { 410 }
68:D      1      Slot11=-20480; { 411 }
69:D      1      Slot12=-16384; { 412 }
70:D      1      Slot13=-12288; { 413 }
71:D      1      Slot14=-8192;  { 414 }
72:S
73:D      1      EventSense1 = Slot0; { Checks pump and switches }
74:D      1      Relay_Out_1 = Slot1; { Controls pump and counters }
75:D      1      D_to_A_1    = Slot2;
76:D      1      A_to_D_1    = Slot4;
77:D      1      Relay_Out_2 = Slot14; { Used to trigger HP-1000 }
78:S
79:D      1 IMPLEMENT
80:C      1 END.
81:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX B.3

CSC-1.3 Pump Commands

```

1:D      0 $UCSD$ $LINES 55$
2:D      0 $SEARCH '#3:PASC_LIB.', '#3:VAR_DEFIN'$
3:S
4:D      0 MODULE PUMP_COMMANDS;
5:S
6:D      1 { Converted for use on the High }
7:D      1 { Strain Wing Repair Tests }
8:S
9:D      1 { Revision dated 24 November 1987 }
10:S
11:S
12:D     1 IMPORT
13:D     1   GENERAL_0, GENERAL_1,
14:D     1   IODECLARATIONS, VAR_DEFINITIONS;
15:S
16:D     1 EXPORT
17:D     1   PROCEDURE POWER_UP;
18:D     1   FUNCTION DUMPED:BOOLEAN;
19:D     1   PROCEDURE DUMP;
20:S
21:D     1 IMPLEMENT
22:S
23:D     1 { ***** POWER_UP ***** }
24:S
25:D     1 PROCEDURE POWER_UP ;
26:C     2 BEGIN
27:C     2   {Zero the load }
28:C     2   WRITEWORD(12,-4000); { DTE,SYE On }
29:C     2   WRITEWORD(12,D_to_A_1);
30:C     2   { Output the voltage }
31:C     2   WRITEWORD(12,-4064); { TME Off, SYE On }
32:C     2   WRITEWORD(12,-4000); { DTE,SYE On }
33:C     2   { Close the relay }
34:C     2   WRITEWORD(12,-4000); { DTE,SYE On }
35:C     2   { Pin 9: (9-1)^2=256 }
36:C     2   WRITEWORD(12,Relay_Out_1+256);
37:C     2 END; { Procedure Power_Up }
38:S
39:S
40:D     1 { ***** DUMPED ***** }
41:S
42:D     1 FUNCTION DUMPED:BOOLEAN;
43:S
44:D     2 VAR
45:D     -4 2   WORD7:INTEGER; { 7 bit word }
46:S
47:C     2 BEGIN
48:C     2   IOCONTROL(12,3,-3792); { IEN,SYE,TME On }
49:C     2   IOCONTROL(12,1,1);
50:C     2   WRITEWORD(12,-3936); { ISL,SYE On }
51:C     2   IOCONTROL(12,3,0);
52:C     2   WORD7:=4095-(IOSTATUS(12,3)+32768); {4095=7777H}
53:C     2   DUMPED:=WORD7<64; { 64=2^6 }
54:C     2 END; { Procedure Dumped }
55:S

```

Pascal [Rev 2.0 10/19/82] PUMP_CMDS.TEXT

24-Nov-87 07:58:14 Page 2

```
56:S
57:D      1  { ***** DUMP ***** }
58:S
59:D      1  PROCEDURE DUMP;
60:C      2  BEGIN
61:C      2      {Open the relay}
62:C      2      IOCONTROL(12,0,1);
63:C      2      WRITEWORD(12,-4000); { DTE,SYE On }
64:C      2      WRITEWORD(12,Relay_Out_1);
65:C      2      {Zero the load output}
66:C      2      WRITEWORD(12,-4000); { DTE,SYE On }
67:C      2      WRITEWORD(12,D_to_A_1);
68:C      2  END; { Proceedure Dump }
69:S
70:C      1  END.
```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX B.4

CSC-1.4 Load Commands

```

1:D      0 $UCSD$ $LINES 55$ $DEBUG ON $
2:D      0 $SEARCH '#3:PASC_LIB.', '#3:VAR_DEFIN'$
3:S
4:D      0 MODULE LOAD_COMMANDS;
5:S
6:D      1 { Converted for use on the High      }
7:D      1 { Strain Wing Repair Tests.          }
8:D      1 { Notes:                             }
9:D      1 {     - 500kip load cell used         }
10:D     1 {     - Command: 10V = LBSMAX        }
11:D     1 {     - Feedback: FS(volts) = 500kips }
12:S
13:D     1 { Revision dated 23 November 1987    }
14:S
15:S
16:D     1 IMPORT
17:D     1   GENERAL_1, IODECLARATIONS,
18:D     1   SYSGLOBALS, CLOCK, VAR_DEFINITIONS;
19:S
20:D     1 EXPORT
21:D     1   PROCEDURE OUTPUT_LOAD (CMDSIG:REAL; LBSMAX:INTEGER);
22:D     1   FUNCTION LOAD (FS:REAL):INTEGER;
23:D     1   PROCEDURE CAL_LOAD (FS:REAL; VAR CAL_DELAY:INTEGER);
24:D     1   PROCEDURE SCAN_1000 (PRINT:BOOLEAN);
25:S
26:D     1 IMPLEMENT
27:S
28:S
29:D     1 { ***** OUTPUT_LOAD ***** }
30:S
31:D     -8 1 PROCEDURE OUTPUT_LOAD (CMDSIG:REAL; LBSMAX:INTEGER);
32:S
33:D     -8 2 VAR
34:D     -12 2   V:INTEGER;
35:S
36:C     2 BEGIN
37*C     2   WRITEWORD(12,-4000); { DTE, SYE On }
38:S
39:C     2   { Convert from OUT_LD to Volts, LSB=5mV }
40:C     2   { [ CMDSIG ] [ 10V ] }
41:C     2   { [-----]*[-----] }
42:C     2   { [ LBSMAX ] [0.005V] }
43*C     2   V:=ROUND(10/0.005*CMDSIG/LBSMAX);
44*C     2   IF V<0 THEN { Correct negative voltage, }
45*C     3     V:=V+4096; { set sign bit, 10 bit word }
46*C     2   V:=V+D_to_A_1; { Correct for slot }
47*C     2   WRITEWORD(12,V); { Output voltage }
48:S
49*C     2 END; { Procedure Output_Load }
50:S
51:S
52:D     1 { ***** LOAD ***** }
53:S
54:D     -8 1 FUNCTION LOAD (FS:REAL):INTEGER;
55:S

```

```

56:D      -8  2  VAR
57:D      -12 2    V:INTEGER;
58:S
59:C      2  BEGIN
60:C      2    { Read voltage }
61*C     2    WRITEWORD(12,-3920); { ISL, SYE, TME On }
62*C     2    WRITEWORD(12,A_to_D_1);
63*C     2    READWORD (12,V);
64:S
65:C     2    { Convert volts to load }
66*C     2    V:=V+32768; { Subtract IRQ (1000000oct) }
67*C     2    IF V>2047 THEN { Correct negative voltage, }
68*C     3      V:=V-4096; { subtract sign bit }
69:C     2    { LSB=5mV for 10V range, }
70:C     2    { FS=500,000lbs, 50/FS = resolution in pounds }
71*C     2    LOAD:=ROUND((0.005*V)*(500000/FS));
72:S
73*C     2  END; { Function Load }
74:S
75:S
76:D     1  { ***** CAL_LOAD ***** }
77:S
78:D     -8  1  PROCEDURE CAL_LOAD (FS:REAL; VAR CAL_DELAY:INTEGER);
79:S
80:D     -8  2  LABEL 10;
81:S
82:D     -8  2  VAR
83:D     -9  2    REPLY:CHAR;
84:D     -18 2    T,TO :INTEGER;
85:D     -22 2    TIME :TIMEREC;
86:S
87:C     2  BEGIN
88*C     2  10: IF CAL_DELAY=0 THEN
89*C     3    CAL_DELAY:=100;
90*C     2    Writeln(FF,TD,TD,TD,LF,LF,LF);
91*C     2    Writeln('          Hit any key to end calibration, or ',LF);
92*C     2    Writeln('          "D" to change cal printout delay');
93*C     2    WRITE  (US,US,US,US,US,US,US,US);
94*C     2    WRITE  ('          Feedback =',LOAD(FS):8,' lbs');
95*C     2    WRITE  (BT,BS,BS,BS,BS);
96:S
97:C     2    REPEAT
98*C     3      WRITE  (LOAD(FS):8,BT);
99*C     3      SYSTIME(TIME);
100*C    3      WITH TIME DO
101*C    4        TO:=HOUR*360000+MINUTE*6000+CENTISECOND;
102:C    3      REPEAT
103*C    4        SYSTIME(TIME);
104*C    4        WITH TIME DO
105*C    5          T:=HOUR*360000+MINUTE*6000+CENTISECOND;
106*C    4        UNTIL (T-TO)>ROUND(CAL_DELAY/10); { delay in millisecond:
3
107*C    3    UNTIL NOT(UNITBUSY(KEYBRD));
108:S
109*C    2    READ(REPLY);
110:C    2

```

```

111*C      2      IF (REPLY='D') OR (REPLY='d') THEN
112:C      3      BEGIN
113*C      3          WRITELN(FF,TD);
114*C      3          WRITELN(' Current delay is ',CAL_DELAY:1,
115:C      3              ' milliseconds',LF);
116*C      3          WRITE (' What is desired delay in milliseconds? ');
117*C      3          READLN(CAL_DELAY);
118*C      3          GOTO 10;
119:C      3      END;
120*C      2 END;  ( Procedure Cal_Load )
121:S
122:S
123:D      1      ( ***** SCAN_1000 ***** )
124:S
125:D      1  PROCEDURE SCAN_1000 (PRINT:BOOLEAN);
126:S
127:D      2  VAR
128:D      -12  2      RELAY2,T0,T:INTEGER;
129:D      -16  2      TIME      :TIMEREC;
130:S
131:C      2  BEGIN
132*C      2      WRITEWORD(12,-4000); ( DTE, SYE On )
133:C      2
134:C      2  { Relay 1 logic "1" triggers scan }
135:C      2  { Relay 2 = "0" for tape file during ramping }
136:C      2  {      = "1" for printout at a hold point }
137:S
138*C      2      IF PRINT THEN
139*C      3          RELAY2:=2 ( Set relay 2 to logic "1" )
140:C      3      ELSE
141*C      3          RELAY2:=0; ( Set relay 2 to logic "0" )
142:S
143*C      2      WRITEWORD(12,Relay_Out_2+1+RELAY2);
144*C      2      SYSTIME(TIME);
145*C      2      WITH TIME DO
146*C      3          T0:=HOUR*360000+MINUTE*6000+CENTISECOND;
147:C      2      REPEAT
148*C      3          SYSTIME(TIME);
149*C      3          WITH TIME DO
150*C      4              T:=HOUR*360000+MINUTE*6000+CENTISECOND;
151*C      3          UNTIL (T-T0)>2; ( 20 millisecond delay )
152*C      2          WRITEWORD(12,Relay_Out_2+0+0)
153:S
154*C      2 END;  ( Procedure Scan_1000 )
155:S
156:C      1 END.
157:S
158:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX B.5

CSC-1.5 Test Parameters

```

1:D      0 $UCSD$ $SEARCH '#3:VAR_DEFIN'$ $LINES 55$
2:D      0 $DEBUG ON $
3:S
4:D      0 MODULE TEST_PARAMETERS;
5:S
6:D      1 { Converted for use on the High }
7:D      1 { Strain Wing Repair Tests }
8:D      1 { Notes: }
9:D      1 { - 500 kip load cell used }
10:D     1 { - Command: 10V = LBSMAX }
11:D     1 { - Feedback: FS(volts) = 500 kips}
12:S
13:D     1 { Revision List: }
14:D     1 { 1.0 23 November 1987 }
15:D     1 { 1.1 4 January 1988 }
16:D     1 { Added Program_End to Export }
17:D     1 { list, which was mistakenly }
18:D     1 { excluded from Version 1.0 }
19:D     1 { 1.2 6 April 1988 }
20:D     1 { Changed Restart to 355 }
21:D     1 { flights of 108 points. }
22:S
23:S
24:D     1 IMPORT
25:D     1 VAR_DEFINITIONS;
26:S
27:D     1 EXPORT
28:S
29:D     1 PROCEDURE CHNG_SYST_PARAM (VAR FS:REAL;
30:D     2 VAR LBSMAX,DLL:INTEGER);
31:D     1 PROCEDURE CHNG_TEST_PARAM (VAR LDRATE,LD_TOL:INTEGER);
32:D     1 PROCEDURE NUMBER_PASSES (VAR TOTPASS:INTEGER);
33:D     1 PROCEDURE RESTART (VAR PASS,FLIGHT,POINT,TOTPASS:INTEGER);
34:D     1 PROCEDURE READ_CONFIG (FATIGUE:BOOLEAN;
35:D     2 VAR FS:REAL; VAR LBSMAX,DLL,LDRATE,LD_TOL,CAL_DELAY,
36:D     2 PASS,FLIGHT,POINT,TOTPASS:INTEGER);
37:D     1 PROCEDURE WRITE_CONFIG (FATIGUE:BOOLEAN; FS:REAL;
38:D     2 LBSMAX,DLL,LDRATE,LD_TOL,CAL_DELAY,
39:D     2 PASS,FLIGHT,POINT,TOTPASS:INTEGER);
40:D     1 FUNCTION PROGRAM_END (FATIGUE:BOOLEAN):BOOLEAN;
41:S
42:D     1 IMPLEMENT
43:S
44:D     1 { ***** CHNG_SYST_PARAM ***** }
45:S
46:D     1 PROCEDURE CHNG_SYST_PARAM (VAR FS:REAL;
47:D     2 VAR LBSMAX,DLL:INTEGER);
48:S
49:D     2 LABEL 10;
50:S
51:D     2 VAR
52:D     -8 2 MAX_LD,TUL:INTEGER;
53:D     -16 2 POUNDS :REAL;
54:D     -17 2 REPLY :CHAR;
55:S

```

```

56:C      2 BEGIN
57*C      2   IF DLL=0 THEN
58:C      3   BEGIN
59*C      3     REPLY:='D';
60*C      3     GOTO 10;
61:C      3   END;
62:S
63*C      2   TUL:=ROUND(1.5*DLL);
64:S
65*C      2   IF LBSMAX=0 THEN
66:C      3   BEGIN
67*C      3     REPLY:='M';
68*C      3     GOTO 10;
69:C      3   END;
70:S
71*C      2   MAX_LD:=ROUND(LBSMAX/TUL*100);
72:S
73:C      2 REPEAT
74:S
75*C      3   WRITELN(FF,LF,LF);
76*C      3   WRITELN(T5,'Design Limit load is ',DLL:6,' lbs');
77*C      3   WRITELN(T5,'Test Ultimate load is ',TUL:6,' lbs',LF);
78:S
79*C      3   WRITELN('Control System:');
80*C      3   WRITELN(T5,'Maximum Load is ',MAX_LD:6,'%TUL');
81*C      3   WRITELN(T5,'Maximum Load is ',LBSMAX:6,' lbs');
82*C      3   WRITELN(T5,'Set Servac Span to ',
83:C      3     ABS(LBSMAX/500000*10):5:2);
84:C      3     {      5 mV      Resolution      }
85:C      3     { Resolution: ----- = ----- }
86:C      3     {      10 V      LBSMAX      }
87*C      3   WRITELN(T5,'Resolution is ',
88:C      3     ABS(ROUND(LBSMAX*0.005/10)):8,' lbs',LF);
89:S
90*C      3   WRITELN('Feedback System:');
91*C      3   WRITELN(T5,'Full scale for 500 kips is ',
92:C      3     FS:6:3,' volts');
93:C      3     {      5 mV      Resolution      }
94:C      3     { Resolution: ----- = ----- }
95:C      3     {      FS      300 kips      }
96*C      3   WRITELN(T5,'Resolution is ',
97:C      3     ABS(ROUND(0.005/FS*500000)):4,' lbs',LF,LF);
98:S
99*C      3   WRITELN(' Hit:');
100*C     3   WRITELN('      D to change Design limit load');
101*C     3   WRITELN('      M to change Maximum actuator load');
102*C     3   WRITELN('      V to change full scale Voltage');
103*C     3   WRITE (' <space> to exit ');
104:C     3 REPEAT
105*C     4     READ(REPLY);
106*C     4     WRITE(BS,SP,BS);
107*C     4   UNTIL (((REPLY='D') OR (REPLY='d')) OR
108:C     4     ((REPLY='M') OR (REPLY='m')) OR
109:C     4     ((REPLY='V') OR (REPLY='v')) OR
110:C     4     (REPLY=' '));

```

```

111:S
112*C      3 10: IF (REPLY='D') OR (REPLY='d') THEN
113:C      4      BEGIN
114*C      4          WRITELN(FF,TD);
115*C      4          IF LBSMAX=0 THEN
116*C      5              REPLY:='M';
117*C      4          WRITE(' Enter Design Limit Load in pounds - ');
118*C      4          READLN(DLL);
119*C      4          DLL:=ABS(DLL);
120*C      4          TUL:=ROUND(1.5*DLL);
121*C      4          MAX_LD:=ROUND(LBSMAX/TUL*100);
122:C      4      END;
123:S
124*C      3      IF (REPLY='M') OR (REPLY='m') THEN
125:C      4      BEGIN
126*C      4          WRITELN(FF,TD);
127*C      4          WRITE(' Enter the maximum test load in pounds - ');
128*C      4          READLN(POUNDS);
129*C      4          LBSMAX:=ROUND(ABS(POUNDS));
130*C      4          MAX_LD:=ROUND(LBSMAX/TUL*100);
131*C      4          IF FS=0 THEN
132*C      5              REPLY:='V';
133:C      4      END;
134:S
135*C      3      IF (REPLY='V') OR (REPLY='v') THEN
136:C      4      BEGIN
137*C      4          WRITELN(FF,TD);
138*C      4          WRITE(' What is corresponding ',
139:C      4              'voltage for 500kips? ');
140*C      4          READLN(FS);
141*C      4          FS:=ABS(FS);
142:C      4      END;
143:C      3
144*C      3      UNTIL REPLY=' ';
145:C      2
146*C      2 END; ( Procedure Chng_Syst_Param )
147:S
148:S
149:S
150:D      1      ( ***** CHNG_TEST_PARAM ***** )
151:S
152:D      1 PROCEDURE CHNG_TEST_PARAM (VAR_LDRATE,LD_TOL:INTEGER);
153:S
154:D      2 LABEL 10;
155:S
156:D      2 VAR
157:D      -4 2      MAX_LD :INTEGER;
158:D      -12 2      POUNDS :REAL;
159:D      -13 2      REPLY :CHAR;
160:S
161:C      2 BEGIN
162*C      2      IF LDRATE=0 THEN
163:C      3      BEGIN
164*C      3          REPLY:='L';
165*C      3          GOTO 10;

```

```

166:C      3      END;
167:S
168:C      2      REPEAT
169:S
170*C      3          WRITELN(FF,TD,TB,'Loading Rate is ',
171:C      3              LDRATE:6,' lbs/sec',LF);
172*C      3          WRITELN(TB,' Tolerance is ',LD_TOL:6,' lbs',TD);
173:S
174*C      3          WRITELN(' Hit:');
175*C      3          WRITELN('          L to change Loading rate');
176*C      3          WRITELN('          T to change Tolerance');
177*C      3          WRITE (' <space> to exit ');
178:C      3      REPEAT
179*C      4          READ(REPLY);
180*C      4          WRITE(BS,SP,BS);
181*C      4          UNTIL (((REPLY='L') OR (REPLY='l')) OR
182:C      4              ((REPLY='T') OR (REPLY='t')) OR
183:C      4              (REPLY=' '));
184:S
185*C      3      10:IF (REPLY='L') OR (REPLY='l') THEN
186:C      4          BEGIN
187*C      4              WRITELN(FF,TD);
188*C      4              WRITE (' What is the ',
189:C      4                  'loading rate in lbs/sec? ');
190*C      4              READLN (LDRATE);
191*C      4              LDRATE:=ABS(LDRATE);
192*C      4              IF LD_TOL=0 THEN
193*C      5                  REPLY:='T';
194:C      4          END;
195:S
196*C      3          IF (REPLY='T') OR (REPLY='t') THEN
197:C      4          BEGIN
198*C      4              WRITELN(FF,TD);
199*C      4              WRITE(' Enter tolerance in pounds - ');
200*C      4              READLN(LD_TOL);
201*C      4              LD_TOL:=ABS(LD_TOL);
202:C      4          END;
203:S
204*C      3      UNTIL REPLY=' ';
205:C      2
206*C      2  END; { Procedure Chng_Test_Param }
207:S
208:S
209:D      1      { ***** NUMBER_PASSES ***** }
210:S
211:D      1  PROCEDURE NUMBER_PASSES (VAR TOTPASS:INTEGER);
212:C      2  BEGIN
213*C      2      WRITELN(FF,TD);
214*C      2      WRITE (' How many passes to be run? ');
215*C      2      READLN (TOTPASS);
216*C      2  END; { Proceedure Number_Passes }
217:S
218:S
219:D      1      { ***** RESTART ***** }
220:S

```

```

221:D      1  PROCEDURE RESTART (VAR PASS,FLIGHT,POINT,TOTPASS: INTEGER);
222:D      2
223:D      2  LABEL 1,2;
224:S
225:D      2  VAR
226:D      -1 2    REPLY:CHAR;
227:S
228:C      2  BEGIN
229*C      2    IF TOTPASS<=0 THEN
230*C      3    NUMBER_PASSES (TOTPASS);
231*C      2    IF (FLIGHT=355) AND (POINT=108) THEN
232:C      3    BEGIN
233*C      3      PASS:=PASS+1;
234*C      3      FLIGHT:=1;
235*C      3      POINT:=1;
236*C      3      GOTO 2;
237:C      3    END;
238:S
239*C      2    IF PASS>0 THEN
240*C      3      GOTO 2;
241:S
242*C      2  1: WRITELN(FF,LF,LF);
243:C      2
244:C      2    { Find Pass number }
245*C      2    IF TOTPASS=1 THEN
246*C      3      PASS:=1
247:C      3    ELSE
248:C      3      REPEAT
249*C      4        WRITELN(TD);
250*C      4        WRITELN(TB,'Enter the Pass number ');
251*C      4        WRITE  (TB,TB,'between 1 and ',TOTPASS:3,' - ');
252*C      4        READ  (PASS);
253*C      4        UNTIL ((PASS>0) AND (PASS<=TOTPASS));
254:C      2
255:C      2    { Find Record number }
256:C      2    REPEAT
257*C      3      WRITELN(LF,TB,'Enter the Flight number');
258*C      3      WRITE  (LF,TB,TB,'between 1 and 355 - ');
259*C      3      READ  (FLIGHT);
260*C      3      UNTIL ((FLIGHT>0) AND (FLIGHT<356));
261:S
262:C      2    { Find Point number }
263:C      2    REPEAT
264*C      3      WRITELN;
265*C      3      WRITELN(TB,'Enter the Point number ');
266*C      3      WRITE  (LF,TB,TB,'between 1 and 108 - ');
267*C      3      READLN (POINT);
268*C      3      UNTIL ((POINT>0) AND (POINT<109));
269:S
270*C      2  2: WRITELN(FF,TD);
271*C      2    WRITELN(TB,'      Pass Number =',PASS:4,LF);
272*C      2    WRITELN(TB,'      Flight Number =',FLIGHT:4,LF);
273*C      2    WRITELN(TB,'      Point Number =',POINT:4,LF);
274*C      2    WRITE  (TB,'Restart at this point? (Y or N) - ');
275:C      2    REPEAT

```

```

276*C      3      READ  (REPLY);
277*C      3      WRITE (BS,SP,BS);
278*C      3      IF (REPLY='N') OR (REPLY='n') THEN
279*C      4      GOTO 1;
280*C      3      UNTIL ((REPLY='Y') OR (REPLY='y'));
281:S
282*C      2      IF PASS>TOTPASS THEN
283*C      3      TOTPASS:=PASS;
284*C      2  END;  { Procedure Restart }
285:S
286:S
287:D      1  { ***** READ_CONFIG ***** }
288:S
289:D      1  PROCEDURE READ_CONFIG (FATIGUE:BOOLEAN;
290:D      2      VAR FS:REAL; VAR LBSMAX,DLL,LDRATE,LD_TOL,CAL_DELAY,
291:D      2      PASS,FLIGHT,POINT,TOTPASS:INTEGER);
292:S
293:D      2  VAR
294:D      -8  2      I,IOR      :INTEGER;
295:D      -9  2      REPLY      :CHAR;
296:D      -52 2      FILENAME,OLDNAME:S20;
297:D      -716 2      CONFIG_FILE :TEXT;
298:S
299:C      2  BEGIN
300*C      2      IF FATIGUE THEN
301*C      3      OLDNAME:='HSWRFCFG'
302:C      3      ELSE
303*C      3      OLDNAME:='HSWRSCFG';
304*C      2      DELETE(FILENAME,1,LENGTH(FILENAME));
305*C      2      WRITELN(FF,TD,TD);
306*C      2      WRITELN(T5,'What is file name to be read?',LF);
307*C      2      IF FATIGUE THEN
308*C      3      WRITE  (T5,'Default is ''HSWRFCFG'' - ');
309:C      3      ELSE
310*C      3      WRITE  (T5,'Default is ''HSWRSCFG'' - ');
311*C      2      READLN (FILENAME);
312*C      2      IF LENGTH(FILENAME)<1 THEN
313*C      3      FILENAME:=OLDNAME;
314*C      2      FILENAME:=CONCAT('#3:',FILENAME);
315*C      2      FILENAME:=CONCAT(FILENAME,'.TEXT');
316:C      2      $IOCHECK OFF$
317*C      2      RESET(CONFIG_FILE,FILENAME);
318:C      2      $IOCHECK ON $
319*C      2      IOR:=IORRESULT;
320*C      2      IF IOR<>0 THEN
321:C      3      BEGIN
322*C      3      WRITELN(TD,T5,'Can't open file ',FILENAME,',',LF);
323*C      3      WRITELN(T5,'IORresult= ',IOR:1,LF);
324*C      3      WRITE  (T5,'Hit any key to continue....');
325*C      3      READ(REPLY);
326:C      3      END
327:C      3      ELSE
328:C      3      BEGIN
329*C      3      READ(CONFIG_FILE,FS,LBSMAX,DLL,
330:C      3      LDRATE,LD_TOL,CAL_DELAY,

```

```

331:C      3      PASS,FLIGHT,POINT,TOTPASS);
332:C      3      CLOSE (CONFIG_FILE);
333:C      3      END;
334:S
335:C      2 END;  ( Procedure Read_Config )
336:S
337:S
338:D      1      ( ***** WRITE_CONFIG ***** )
339:S
340:D      1 PROCEDURE WRITE_CONFIG (FATIGUE:BOOLEAN; FS:REAL;
341:D      2      LBSMAX,DLL,LDRATE,LD_TOL,CAL_DELAY,
342:D      -8  2      PASS,FLIGHT,POINT,TOTPASS:INTEGER);
343:S
344:D      -8  2 VAR
345:D      -12 2      IOR          :INTEGER;
346:D      -13 2      REPLY        :CHAR;
347:D      -56 2      FILENAME,OLDNAME:S20;
348:D      -720 2      CONFIG_FILE :TEXT;
349:S
350:C      2 BEGIN
351:C      2      IF FATIGUE THEN
352:C      3          OLDNAME:='HSWRFCFG'
353:C      3      ELSE
354:C      3          OLDNAME:='HSWRSCFG';
355:C      2      DELETE (FILENAME,1,LENGTH (FILENAME));
356:C      2      WRITELN (FF,TD,TD);
357:C      2      WRITELN (T5,'What is file name to be written?',LF);
358:C      2      IF FATIGUE THEN
359:C      3          WRITE  (T5,'Default is ''HSWRFCFG'' - ');
360:C      3      ELSE
361:C      3          WRITE  (T5,'Default is ''HSWRSCFG'' - ');
362:C      2      READLN (FILENAME);
363:C      2      IF LENGTH (FILENAME)<1 THEN
364:C      3          FILENAME:=OLDNAME;
365:C      2      FILENAME:=CONCAT ('#3:',FILENAME);
366:C      2      FILENAME:=CONCAT (FILENAME,'.TEXT');
367:C      2      $IOCHECK OFF;
368:C      2      REWRITE (CONFIG_FILE,FILENAME);
369:C      2      $IOCHECK ON;
370:C      2      IOR:=IORRESULT;
371:C      2      IF IOR<>0 THEN
372:C      3          BEGIN
373:C      3              WRITELN (TD,T5,'Can''t open file ',FILENAME,',',LF);
374:C      3              WRITELN (T5,'IORresult= ',IOR:1,LF);
375:C      3              WRITE  (T5,'Hit any key to continue....');
376:C      3              READ (REPLY);
377:C      3          END
378:C      3      ELSE
379:C      3          BEGIN
380:C      3              WRITELN (CONFIG_FILE,FS,LBSMAX,DLL,
381:C      3                  LDRATE,LD_TOL,CAL_DELAY,
382:C      3                  PASS,FLIGHT,POINT,TOTPASS);
383:C      3              CLOSE  (CONFIG_FILE,'SAVE');
384:C      3          END;
385:S

```



```

386*C      2 END; { Procedure Write_Config }
387:S
388:S
389:D      1 { ***** PROGRAM_END ***** }
390:S
391:D      1 FUNCTION PROGRAM_END (FATIGUE:BOOLEAN):BOOLEAN;
392:S
393:D      2 VAR
394:D      -1 2   REPLY:CHAR;
395:S
396:C      2 BEGIN
397*C      2   WRITELN(BL,FF,TD);
398*C      2   WRITELN(' Are you sure you want to end the program?');
399*C      2   IF FATIGUE THEN
400*C      3     WRITE(' Configuration changes and/or the fatigue',
401:C      3       CR,LF,' test counters')
402:C      3   ELSE
403*C      3     WRITE(' Configuration changes');
404*C      2   WRITELN(' will be lost....',TD);
405*C      2   WRITE (' End the program(Y or N)? ');
406:C      2   REPEAT
407*C      3     READ (REPLY);
408*C      3     WRITE (BS,SP,BS);
409*C      3   UNTIL (REPLY='Y') OR (REPLY='y') OR
410:C      3     (REPLY='N') OR (REPLY='n');
411:S
412*C      2   PROGRAM_END:= ((REPLY='Y') OR (REPLY='y'));
413:S
414*C      2 END; { Function Program_End }
415:S
416:S
417:C      1 END. { Test_Parameters }
418:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX B.6

CSC-1.6 Test Conditions

```

1:D      0 $UCSD$ $LINES 55$
2:D      0 $SEARCH '#47:PASC_LIB.', '#3:VAR_DEFIN'$
3:S
4:D      0 MODULE TEST_CONDITIONS;
5:S
6:D      1 { Converted for use on the High }
7:D      1 { Strain Wing Repair Tests }
8:S
9:D      1 { Revision dated 16 November 1987 }
10:D     1 { for Switch_Set }
11:D     1 { Revision dated 16 December 1987 }
12:D     1 { for remaining procedures }
13:S
14:D     1 IMPORT
15:D     1 GENERAL_0, GENERAL_1,
16:D     1 IODECLARATIONS, VAR_DEFINITIONS;
17:S
18:D     1 EXPORT
19:S
20:D     1 FUNCTION SWITCH_SET (SWITCH_NO: INTEGER): BOOLEAN;
21:D     1 PROCEDURE CLOSE_RELAY (COUNTER: INTEGER);
22:D     1 PROCEDURE OPEN_RELAY;
23:D     1 PROCEDURE ZERO_COUNTERS;
24:D     1 PROCEDURE RESET_COUNTERS (PASS, RECD, POINT: INTEGER);
25:D     1 PROCEDURE INCR_COUNTER (COUNTER: INTEGER);
26:S
27:D     1 IMPLEMENT
28:S
29:S
30:D     1 { ***** SWITCH_SET ***** }
31:S
32:D     1 FUNCTION SWITCH_SET (SWITCH_NO: INTEGER): BOOLEAN;
33:D     2 { This function returns value of True if the switch }
34:D     2 { number on Event Sense card SWITCH_NO is set
35:D     2 { Notes: }
36:D     2 { - Pin no. on card = SWITCH_NO + 1 }
37:D     2 { - Exp(Ln(2)*I)=2^I }
38:D     2 { - Pump interlock is Switch #6 }
39:S
40:D     2 VAR
41:D     -8 2 I, WORD7: INTEGER;
42:D     -16 2 Ln2: REAL;
43:S
44:C     2 BEGIN
45:C     2 Ln2:=Ln(2);
46:C     2 IOCONTROL(12,3,-3792); { IEN, SYE, TME On }
47:C     2 IOCONTROL(12,1,1);
48:C     2 WRITEWORD(12,-3936);
49:C     2 IOCONTROL(12,3,0);
50:C     2 WORD7:=4095-(IOSTATUS(12,3)+32768); {4095=7777H}
51:C     2 FOR I:=6 DOWNT0 SWITCH_NO+1 DO
52:C     3 IF WORD7>=Exp(Ln2*I) THEN {2^I}
53:C     4 WORD7:=WORD7-ROUND(Exp(Ln2*I)); {2^I}
54:C     2 SWITCH_SET:=WORD7>=Exp(Ln2*SWITCH_NO); {2^SWITCH_NO}
55:C     2 END; { Function Set_Switch }

```

```

56:S
57:S
58:D      1 { ***** CLOSE_RELAY ***** }
59:S
60:D      1 PROCEDURE CLOSE_RELAY (COUNTER:INTEGER);
61:C      2 BEGIN
62:C      2     { Find bit corresponding to COUNTER }
63:C      2     IF COUNTER<0 THEN
64:C      3       COUNTER:=-COUNTER*2-1
65:C      3     ELSE
66:C      3       COUNTER:=(COUNTER-1)*2;
67:C      2     COUNTER:=ROUND(Exp(Ln(2)*COUNTER)); {2^I}
68:C      2     { Close the relay, but don't dump pump }
69:C      2     { Pump is 7th relay, 100 hex = 256 }
70:C      2     WRITEWORD(12,-4000);
71:C      2     WRITEWORD(12,Relay_Out_1+COUNTER+256);
72:C      2 END;   { Procedure Close_Relay }
73:S
74:S
75:D      1 { ***** OPEN_RELAY ***** }
76:S
77:D      1 PROCEDURE OPEN_RELAY;
78:C      2 BEGIN
79:C      2     { Open the relay, but don't dump pump }
80:C      2     WRITEWORD(12,-4000);
81:C      2     WRITEWORD(12,Relay_Out_1+256);
82:C      2 END;   { Procedure Close_Relay }
83:S
84:S
85:D      1 { ***** ZERO_COUNTERS ***** }
86:S
87:D      1 PROCEDURE ZERO_COUNTERS;
88:S
89:D      2 VAR
90:D      -1 2     DUMMY:BOOLEAN;
91:S
92:C      2 BEGIN
93:C      2     { Close the relays, but don't dump pump }
94:C      2     WRITEWORD(12,-4000);
95:C      2     WRITEWORD(12,Relay_Out_1+170+256); { 170=AA hex }
96:C      2     DUMMY:=SWITCH_SET(7);
97:C      2     OPEN_RELAY;
98:C      2 END;   { Procedure Zero_Counters }
99:S
100:S
101:D      1 { ***** RESET_COUNTERS ***** }
102:S
103:D      1 PROCEDURE RESET_COUNTERS (PASS,RECRD,POINT:INTEGER);
104:S
105:D      2 VAR
106:D      -4 2     I:INTEGER;
107:D      -5 2     DUMMY:BOOLEAN; { Used as program delay as }
108:D      -5 2     { required by counters }
109:S
110:C      2 BEGIN

```

```

111:C      2      { Zero pass counter }
112:C      2      DUMMY:=SWITCH_SET(7);
113:C      2      CLOSE_RELAY (-3);
114:C      2      DUMMY:=SWITCH_SET(7); DUMMY:=SWITCH_SET(7);
115:C      2      OPEN_RELAY;
116:C      2      { Reset pass counter }
117:C      2      FOR I:=1 TO PASS DO
118:C      3      BEGIN
119:C      3          DUMMY:=SWITCH_SET(7);
120:C      3          CLOSE_RELAY (3);
121:C      3          DUMMY:=SWITCH_SET(7);
122:C      3          OPEN_RELAY;
123:C      3      END;
124:C      2      DUMMY:=SWITCH_SET(7);
125:S
126:C      2      { Reset record counter }
127:C      2      FOR I:=1 TO RECRD DO
128:C      3      BEGIN
129:C      3          DUMMY:=SWITCH_SET(7);
130:C      3          CLOSE_RELAY (2);
131:C      3          DUMMY:=SWITCH_SET(7);
132:C      3          OPEN_RELAY;
133:C      3      END;
134:S
135:C      2      { Reset point counter }
136:C      2      FOR I:=1 TO POINT DO
137:C      3      BEGIN
138:C      3          DUMMY:=SWITCH_SET(7);
139:C      3          CLOSE_RELAY (1);
140:C      3          DUMMY:=SWITCH_SET(7);
141:C      3          OPEN_RELAY;
142:C      3      END;
143:C      2      END; { Procedure Reset_Counters }
144:S
145:S
146:D      1 { ***** INCR_COUNTER ***** }
147:S
148:D      1 PROCEDURE INCR_COUNTER (COUNTER:INTEGER);
149:S
150:D      2 VAR
151:D      -1 2      DUMMY:BOOLEAN;
152:S
153:C      2 BEGIN
154:C      2          CLOSE_RELAY (COUNTER);
155:C      2          DUMMY:=SWITCH_SET(7); { Program delay }
156:C      2          OPEN_RELAY;
157:C      2      END; { Procedure Incr_Counter }
158:S
159:C      1      END.
160:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX C

CSCI-2 PROGRAM LISTINGS FOR VERIFICATION PROGRAMS

APPENDIX C.1

CSC-2.1 Check Pump Commands

```

1:D      0 $UCSD$ $LINES 55$ $DEBUG ON$
2:D      0 $SEARCH '#47:PASC_LIB.' , '#3:VAR_DEFIN'$
3:S
4:D      0 PROGRAM HSWR_CHECK_PUMP_CMDS;
5:D      1
6:D      1 { This program checks the Power_Up }
7:D      1 { and Dump Procedures }
8:D      1 { Notes: }
9:D      1 { - PASC_LIB and VAR_DEFIN required }
10:D     1 { - Jumper required from Event Sense }
11:D     1 { jack to Relay Output jack on }
12:D     1 { counter panel }
13:S
14:D     1 { Revision dated 24 November 1987 }
15:S
16:D     1 LABEL 10;
17:S
18:D     1 IMPORT
19:D     1 IODECLARATIONS,GENERAL_0,
20:D     1 GENERAL_1,VAR_DEFINITIONS;
21:S
22:D     1 CONST
23:D     1 ON ='On ';
24:D     1 OFF='Off';
25:S
26:D     1 VAR
27:D     -8 1 I,SWITCH_NO:INTEGER;
28:D     -9 1 REPLY:CHAR;
29:S
30:D     -9 1 { MODULE PUMP_COMMANDS }
31:S
32:D     -9 1 { ***** POWER_UP ***** }
33:S
34:D     1 PROCEDURE POWER_UP ;
35:C     2 BEGIN
36:C     2 {Zero the load }
37:C     2 WRITEWORD(12,-4000); { DTE,SYE On }
38:C     2 WRITEWORD(12,D_to_A_1);
39:C     2 { Output the voltage }
40:C     2 WRITEWORD(12,-4064); { TME Off, SYE On }
41:C     2 WRITEWORD(12,-4000); { DTE,SYE On }
42:C     2 { Close the relay }
43:C     2 WRITEWORD(12,-4000); { DTE,SYE On }
44:C     2 { Pin 9: (9-1)^2=256 }
45:C     2 WRITEWORD(12,Relay_Out_1+256);
46:C     2 END; { Proceedure Power_Up }
47:S
48:S
49:D     -9 1 { ***** DUMPED ***** }
50:S
51:D     1 FUNCTION DUMPED:BOOLEAN;
52:S
53:D     2 VAR
54:D     -4 2 WORD7:INTEGER; { 7 bit word }
55:S

```



```

56:C      2 BEGIN
57*C      2   IOCONTROL(12,3,-3792); { IEN,SYE,TME On }
58*C      2   IOCONTROL(12,1,1);
59*C      2   WRITEWORD(12,-3936); { ISL,SYE On }
60*C      2   IOCONTROL(12,3,0);
61*C      2   WORD7:=4095-(IOSTATUS(12,3)+32768); {4095=7777H}
62*C      2   DUMPED:=WORD7<64; { 64=2^6 }
63*C      2 END; { Procedure Dumped }
64:S
65:S
66:D      -9  1   { ***** DUMP ***** }
67:S
68:D      1 PROCEDURE DUMP;
69:C      2 BEGIN
70:C      2   {Open the relay}
71*C      2   IOCONTROL(12,0,1);
72*C      2   WRITEWORD(12,-4000); { DTE,SYE On }
73*C      2   WRITEWORD(12,Relay_Out_1);
74:C      2   {Zero the load output}
75*C      2   WRITEWORD(12,-4000); { DTE,SYE On }
76*C      2   WRITEWORD(12,D_to_A_1);
77*C      2 END; { Procedure Dump }
78:S
79:S
80:C      1 BEGIN
81*C      1   IOINITIALIZE;
82*C      1   WRITELN(LF,LF,LF,LF,LF,LF,LF,LF,LF);
83*C      1   WRITELN(LF,LF,LF,LF,LF,LF,TB,'Hit any key to quit, or');
84*C      1   WRITELN(TB,T5,'P' to Power up pump');
85*C      1   WRITELN(TB,T5,'D' to Dump pump');
86*C      1   WRITELN(US,US,US,US,US,US,US,US,US,US,US);
87*C      1   WRITE (T5,'Pump is ');
88:C      1 10: REPEAT
89*C      2   IF DUMPED THEN
90*C      3   WRITE (BS,BS,BS,OFF)
91:C      3   ELSE
92*C      3   WRITE (BS,BS,BS,ON);
93:S
94*C      2   UNTIL NOT(UNITBUSY(KEYBRD));
95*C      1   READ(REPLY);
96*C      1   WRITE(BS,SP,BS);
97:S
98*C      1   IF (REPLY='P') OR (REPLY='p') THEN
99:C      2   BEGIN
100*C     2   POWER_UP;
101*C     2   GOTO 10;
102:C     2   END;
103:S
104*C     1   IF (REPLY='D') OR (REPLY='d') THEN
105:C     2   BEGIN
106*C     2   DUMP;
107*C     2   GOTO 10;
108:C     2   END;
109:S
110*C     1   WRITELN(LF,LF,LF,LF,LF,LF);

```

Pascal [Rev 2.0 10/19/82] HSWRPC.TEXT

24-Nov-87 07:51:28 Page 3

```
111*C      1    WRITELN(TB,'Program ended....',TB);
112*C      1    WRITELN(TB,TB,TB,TB);
113*C      1    WRITELN(TB,TB,TB,TB);
114*C      1    IOUNINITIALIZE;
115*C      1    END.
116:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX C.2

CSC-2.2 Check Load Commands

```

1:D      0 $UCSD$ $LINES 55$ $DEBUG ON$
2:D      0 $SEARCH '#47:PASC_LIB.', '#3:VAR_DEFIN'$
3:S
4:D      0 PROGRAM HSWR_CHECK_LOAD_CMDS;
5:S
6:D      1
7:D      1 { This program checks the LOAD_CMDS }
8:D      1 { Module. }
9:D      1 { Notes: }
10:D     1 { - PASC_LIB and VAR_DEFIN required }
11:D     1 { - Jumper required from the D-to-A }
12:D     1 { card to the A-to-D card. }
13:S
14:D     1 { Revision dated 23 November 1987 }
15:S
16:D     1 LABEL 10,20,25,30;
17:S
18:D     1 IMPORT
19:D     1 IODECLARATIONS,GENERAL_0,GENERAL_1,
20:D     1 CLOCK,SYSGLOBALS,VAR_DEFINITIONS;
21:S
22:D     1 VAR
23:D     1 CAL_DELAY, { Time delay for screen update }
24:D     1 NEXTLD, { Load input by operator }
25:D     1 LASTLD, { Previous load point }
26:D     1 LBSMAX { Max actuator load in pounds, span setting }
27:D    -16 1 { Loads are integer }:INTEGER;
28:D    -20 1 TIME :TIMEREC;
29:D    -20 1 TO,T,FS,{ Start/Elapse time, feedback full scale in volts :
30:D    -20 1 POUNDS, { Used for inputting NEXTLD }
31:D    -20 1 CMDSIG, { Command Signal in pounds }
32:D    -76 1 OMEGA,SIN_TERM:REAL;
33:D    -77 1 REPLY :CHAR;
34:S
35:D    -77 1 { ***** MODULE LOAD_COMMANDS ***** }
36:S
37:D    -77 1 { ***** OUTPUT_LOAD ***** }
38:S
39:D     -8 1 PROCEDURE OUTPUT_LOAD (CMDSIG:REAL; LBSMAX:INTEGER);
40:S
41:D     -8 2 VAR
42:D    -12 2 V:INTEGER;
43:S
44:C      2 BEGIN
45:C      2 WRITEWORD(12,-4000); { DTE, SYE On }
46:S
47:C      2 { Convert from OUT_LD to Volts, LSB=5mV }
48:C      2 { [ CMDSIG ] [ 10V ] }
49:C      2 { [-----]*[-----] }
50:C      2 { [ LBSMAX ] [0.005V] }
51:C      2 V:=ROUND(10/0.005*CMDSIG/LBSMAX);
52:C      2 IF V<0 THEN { Correct negative voltage, }
53:C      3 V:=V+4096; { set sign bit, 10 bit word }
54:C      2 V:=V+D_to_A_1; { Correct for slot }
55:C      2 WRITEWORD(12,V); { Output voltage }

```

```

56:S
57*C      2 END;  { Proceedure Output_Load }
58:S
59:S
60:D      -77  1  { ***** LOAD ***** }
61:S
62:D      -8   1 FUNCTION LOAD (FS:REAL):INTEGER;
63:S
64:D      -8   2 VAR
65:D      -12  2   V: INTEGER;
66:S
67:C      2 BEGIN
68:C      2   { Read voltage }
69*C      2   WRITEWORD(12,-3920); { ISL, SYE, TME On }
70*C      2   WRITEWORD(12,A_to_D_1);
71*C      2   READWORD (12,V);
72:S
73:C      2   { Convert volts to load }
74*C      2   V:=V+32768; { Subtract IRQ (100000oct) }
75*C      2   IF V>2047 THEN { Correct negative voltage, }
76*C      3     V:=V-4096; { subtract sign bit }
77:C      2   { LSB=5mV for 10V range, }
78:C      2   { FS=500,000lbs, 50/FS = resolution in pounds }
79*C      2   LOAD:=ROUND((0.005*V)*(500000/FS));
80:S
81*C      2 END;  { Function Load }
82:S
83:S
84:D      -77  1  { ***** CAL_LOAD ***** }
85:S
86:D      -8   1 PROCEDURE CAL_LOAD (FS:REAL; VAR CAL_DELAY:INTEGER);
87:S
88:D      -8   2 LABEL 10;
89:S
90:D      -8   2 VAR
91:D      -9   2   REPLY:CHAR;
92:D      -18  2   T,TO :INTEGER;
93:D      -22  2   TIME :TIMEREC;
94:S
95:C      2 BEGIN
96*C      2 10:IF CAL_DELAY=0 THEN
97*C      3   CAL_DELAY:=100;
98*C      2   WRITELN(FF,TD,TD,TD,LF,LF,LF);
99*C      2   WRITELN('          Hit any key to end calibration, or ',LF);
100*C     2   WRITELN('          "D" to change cal printout delay');
101*C     2   WRITE  (US,US,US,US,US,US,US,US);
102*C     2   WRITE  ('          Feedback =',LOAD(FS):B,' lbs');
103*C     2   WRITE  (BT,BS,BS,BS,BS);
104:S
105:C     2   REPEAT
106*C     3     WRITE  (LOAD(FS):B,BT);
107*C     3     SYTIME(TIME);
108*C     3     WITH TIME DO
109*C     4       TO:=HOUR*360000+MINUTE*6000+CENTISECOND;
110:C     3     REPEAT

```

Pascal [Rev 2.0 10/19/82] HSWRLC.TEXT

23-Nov-87 12:36:17 Page 3

```

111*C      4      SYSTIME(TIME);
112*C      4      WITH TIME DO
113*C      5          T:=HOUR*360000+MINUTE*6000+CENTISECOND;
114*C      4      UNTIL (T-T0)>ROUND(CAL_DELAY/10); { delay in millisecond:
)
115*C      3      UNTIL NOT(UNITBUSY(KEYBRD));
116:S
117*C      2      READ(REPLY);
118:C
119*C      2      IF (REPLY='D') OR (REPLY='d') THEN
120:C      3      BEGIN
121*C      3          WRITELN(FF,TD);
122*C      3          WRITELN(' Current delay is ',CAL_DELAY:1,
123:C      3              ' milliseconds',LF);
124*C      3          WRITE (' What is desired delay in milliseconds? ');
125*C      3          READLN(CAL_DELAY);
126*C      3          GOTO 10;
127:C      3      END;
128*C      2 END; { Procedure Cal_Load }
129:S
130:S
131:D      -77  1      { ***** SCAN_1000 ***** }
132:S
133:D      1  PROCEDURE SCAN_1000 (PRINT:BOOLEAN);
134:S
135:D      2  VAR
136:D      -12  2      RELAY2,T0,T:INTEGER;
137:D      -16  2      TIME      :TIMEREC;
138:S
139:C      2  BEGIN
140*C      2      WRITEWORD(12,-4000); { DTE, SYE On }
141:C
142:C      2  { Relay 1 logic "1" triggers scan }
143:C      2  { Relay 2 = "0" for tape file during ramping }
144:C      2  {      = "1" for printout at a hold point }
145:S
146*C      2      IF PRINT THEN
147*C      3          RELAY2:=2 { Set relay 2 to logic "1" }
148:C      3      ELSE
149*C      3          RELAY2:=0; { Set relay 2 to logic "0" }
150:S
151*C      2      WRITEWORD(12,Relay_Out_2+1+RELAY2);
152*C      2      SYSTIME(TIME);
153*C      2      WITH TIME DO
154*C      3          T0:=HOUR*360000+MINUTE*6000+CENTISECOND;
155:C      2      REPEAT
156*C      3          SYSTIME(TIME);
157*C      3          WITH TIME DO
158*C      4              T:=HOUR*360000+MINUTE*6000+CENTISECOND;
159*C      3          UNTIL (T-T0)>2; { 20 millisecond delay }
160*C      2          WRITEWORD(12,Relay_Out_2+0+0)
161:S
162*C      2 END; { Procedure Scan_1000 }
163:S
164:S
165:S

```

```

166:S
167:C      1 BEGIN
168*C      1   IDINITIALIZE;
169*C      1   LASTLD:=0;
170*C      1   NEXTLD:=0;
171*C      1   CMDSIG:=0;
172*C      1   OMEGA:=0.1; { 10 second period }
173*C      1   OUTPUT_LOAD(0,500000); { Zero output }
174:S
175*C      1   IF LBSMAX=0 THEN
176:C      2   BEGIN
177*C      2     REPLY:='M';
178*C      2     GOTO 30;
179:C      2   END;
180:S
181*C      1 10:WRITELN(FF,TD);
182*C      1   WRITELN('Command: Maximum Load is ',LBSMAX:6,' lbs',LF);
183*C      1   WRITELN('Feedback: Full scale for 500 kips is ',
184:C      1     FS:6:3,' volts',TD,LF,LF,LF);
185*C      1   WRITELN(T5,'Enter Output value in pounds, ',LF);
186*C      1   WRITELN(T5,'or -99 to stop ');
187:S
188*C      1 20:WRITELN(TU,TU,TB,' Command =',CMDSIG:8:0,' lbs',LF);
189*C      1   WRITE  (TB,'Feedback =',LOAD(FS):8,' lbs',
190:C      1     BS,BS,BS,BS);
191:S
192*C      1   IF NEXTLD=LASTLD THEN
193*C      2     GOTO 25;
194:S
195:C      1   { Find starting time }
196*C      1   SYSTIME(TIME);
197*C      1   WITH TIME DO
198*C      2     TO:=HOUR*3600+MINUTE*60+CENTISECOND/100;
199:S
200:C      1   REPEAT
201:C      2     { Determine value of next load step }
202*C      2     SYSTIME(TIME);
203*C      2     WITH TIME DO
204*C      3     T:=HOUR*3600+MINUTE*60+CENTISECOND/100-TO;
205*C      2     IF (OMEGA*T)<0.5 THEN
206*C      3     SIN_TERM:=0.5*(SIN(PI*(2*OMEGA*T-0.5))+1)
207:C      3     ELSE
208*C      3     SIN_TERM:=1.0;
209:S
210:C      2     { Step up loads, read/display loads }
211*C      2     CMDSIG:=(NEXTLD-LASTLD)*SIN_TERM+LASTLD;
212*C      2     WRITE(BT,US,US,CMDSIG:8:0);
213*C      2     OUTPUT_LOAD(CMDSIG,LBSMAX);
214*C      2     WRITE  (BT,LF,LF,LOAD(FS):8);
215:S
216*C      2   UNTIL SIN_TERM=1.0; { Reached load point? }
217:S
218:C      1   { Hold at load point until operator input }
219:C      1 25:REPEAT
220*C      2     WRITE  (BT,LOAD(FS):8);

```

```

221*C      2      UNTIL NOT(UNITBUSY(KEYBRD));
222:S
223*C      1      WRITE (CR,TD,LF,T5,'or -99 to stop ');
224*C      1      LASTLD:=NEXTLD;
225*C      1      READ(NEXTLD);
226*C      1      IF NEXTLD=LASTLD THEN
227:C      2      BEGIN
228*C      2          WRITELN(US,TB,TB,TB,TB,CR,T5,'or -99 to stop ');
229*C      2          WRITE (TU,US,US,TB,'Feedback =',LOAD(FS):8);
230*C      2          GOTO 25;
231:C      2      END;
232:S
233*C      1      IF NEXTLD<>-99 THEN
234*C      2          GOTO 10;
235:S
236*C      1      LASTLD:=0;
237*C      1      NEXTLD:=0;
238*C      1      CMDSIG:=0;
239*C      1      OUTPUT_LOAD(CMDSIG,LBSMAX);
240:S
241*C      1      WRITELN(US,US,US,T5,
242:C      1          'What would you like to do now? Hit:',LF);
243*C      1      WRITELN(TB,'M to change Maximum actuator load');
244*C      1      WRITELN(TB,'V to change full scale Voltage');
245*C      1      WRITELN(TB,'C to run Calibration routine');
246*C      1      WRITELN(TB,'S to sent Scan signal to HP-1000');
247*C      1      WRITELN(TB,'P to sent Print signal to HP-1000');
248*C      1      WRITE (TB,'Q to Quit ');
249:C      1      REPEAT
250*C      2          READ(REPLY);
251*C      2          WRITE(BS,SP,BS);
252*C      2          UNTIL (((REPLY='M') OR (REPLY='m')) OR
253:C      2              ((REPLY='V') OR (REPLY='v')) OR
254:C      2              ((REPLY='C') OR (REPLY='c')) OR
255:C      2              ((REPLY='S') OR (REPLY='s')) OR
256:C      2              ((REPLY='P') OR (REPLY='p')) OR
257:C      2              ((REPLY='Q') OR (REPLY='q')));
258:S
259*C      1      30: IF (REPLY='M') OR (REPLY='m') THEN
260:C      2          BEGIN
261*C      2              WRITELN(FF,TD);
262*C      2              WRITE(' Enter the maximum test load in pounds - ');
263*C      2              READLN(POUNDS);
264*C      2              LBSMAX:=ROUND(ABS(POUNDS));
265*C      2              IF FS=0 THEN
266*C      3                  REPLY:='V';
267:C      2          END;
268:S
269*C      1      IF (REPLY='V') OR (REPLY='v') THEN
270:C      2          BEGIN
271*C      2              WRITELN(FF,TD);
272*C      2              WRITE (' What is corresponding ',
273:C      2                  'voltage for 500 kips? ');
274*C      2              READLN(FS);
275*C      2              FS:=ABS(FS);

```


Pascal [Rev 2.0 10/19/82] HSWRLC.TEXT

23-Nov-87 12:36:17 Page 6

```
276:C      2      END;
277:S
278*C      1      IF (REPLY='C') OR (REPLY='c') THEN
279*C      2          CAL_LOAD (FS,CAL_DELAY);
280:C      1
281*C      1      IF (REPLY='S') OR (REPLY='s') THEN
282*C      2          SCAN_1000 (FALSE);
283:S
284*C      1      IF (REPLY='P') OR (REPLY='p') THEN
285*C      2          SCAN_1000 (TRUE);
286:S
287*C      1      IF NOT((REPLY='Q') OR (REPLY='q')) THEN
288*C      2          GOTO 10;
289:S
290*C      1      Writeln(LF,LF);
291*C      1      Writeln(TB,'Program ended....');
292*C      1      IOUNINITIALIZE;
293*C      1      END.
294:S
295:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX C.3

CSC-2.3 Check Test Parameter Routines

```

1:D      0 $UCSD$ $LINES 55$ $DEBUG ON$ $LIST ON $
2:D      0 $SEARCH '#3:VAR_DEFIN'$ $REF 50$
3:S
4:D      0 PROGRAM HSWR_CHECK_TEST_PARM;
5:D      1
6:D      1      { This program checks the routines      }
7:D      1      { in TEST_PARM.                          }
8:D      1      { Notes:                                  }
9:D      1      { - PASC_LIB and VAR_DEFIN required      }
10:S
11:D     1      { Revision dated 23 November 1987      }
12:S
13:D     1 LABEL 10,90;
14:S
15:D     1 IMPORT
16:D     1     VAR_DEFINITIONS;
17:S
18:D     1 VAR
19:D     1     I, SWITCH_NO,
20:D     1     LBSMAX, DLL, LDRATE,
21:D     1     LD_TOL, CAL_DELAY,
22:D     1     PASS, RECRD, POINT,
23:D    -44 1     TOTPASS           : INTEGER;
24:D    -45 1     REPLY             : CHAR;
25:D    -47 1     FATIGUE, CONFIG_CHNG: BOOLEAN;
26:D    -56 1     FS                : REAL;
27:S
28:S
29:S
30:D    -56 1      { ***** CHNG_SYST_PARAM ***** }
31:S
32:D     1 PROCEDURE CHNG_SYST_PARAM (VAR FS:REAL;
33:D     2     VAR LBSMAX, DLL: INTEGER);
34:S
35:D     2 LABEL 10;
36:S
37:D     2 VAR
38:D    -8  2     MAX_LD, TUL: INTEGER;
39:D   -16  2     POUNDS      : REAL;
40:D   -17  2     REPLY       : CHAR;
41:S
42:C     2 BEGIN
43:C     2     IF DLL=0 THEN
44:C     3     BEGIN
45:C     3         REPLY:='D';
46:C     3         GOTO 10;
47:C     3     END;
48:S
49:C     2     TUL:=ROUND(1.5*DLL);
50:S
51:C     2     IF LBSMAX=0 THEN
52:C     3     BEGIN
53:C     3         REPLY:='M';
54:C     3         GOTO 10;
55:C     3     END;

```

```

56:S
57*C      2      MAX_LD:=ROUND(LBSMAX/TUL*100);
58:S
59:C      2      REPEAT
60:S
61*C      3      WRITELN(FF,LF,LF);
62*C      3      WRITELN(T5,'Design Limit load is ',DLL:6,' lbs');
63*C      3      WRITELN(T5,'Test Ultimate load is ',TUL:6,' lbs',LF);
64:S
65*C      3      WRITELN('Control System:');
66*C      3      WRITELN(T5,'Maximum Load is ',MAX_LD:6,'%TUL');
67*C      3      WRITELN(T5,'Maximum Load is ',LBSMAX:6,' lbs');
68*C      3      WRITELN(T5,'Set Servac Span to ',
69:C      3      ABS(LBSMAX/500000*10):5:2);
70:C      3      {          5 mV      Resolution  }
71:C      3      { Resolution: ----- = ----- }
72:C      3      {          10 V      LBSMAX      }
73*C      3      WRITELN(T5,'Resolution is ',
74:C      3      ABS(ROUND(LBSMAX*0.005/10)):8,' lbs',LF);
75:S
76*C      3      WRITELN('Feedback System:');
77*C      3      WRITELN(T5,'Full scale for 500 kips is ',
78:C      3      FS:6:3,' volts');
79:C      3      {          5 mV      Resolution  }
80:C      3      { Resolution: ----- = ----- }
81:C      3      {          FS        300 kips    }
82*C      3      WRITELN(T5,'Resolution is ',
83:C      3      ABS(ROUND(0.005/FS*500000)):4,' lbs',LF,LF);
84:S
85*C      3      WRITELN(' Hit:');
86*C      3      WRITELN('      D to change Design limit load');
87*C      3      WRITELN('      M to change Maximum actuator load');
88*C      3      WRITELN('      V to change full scale Voltage');
89*C      3      WRITE (' <space> to exit ');
90:C      3      REPEAT
91*C      4      READ(REPLY);
92*C      4      WRITE(BS,SP,BS);
93*C      4      UNTIL (((REPLY='D') OR (REPLY='d')) OR
94:C      4      ((REPLY='M') OR (REPLY='m')) OR
95:C      4      ((REPLY='V') OR (REPLY='v')) OR
96:C      4      (REPLY=' '));
97:S
98*C      3 10: IF (REPLY='D') OR (REPLY='d') THEN
99:C      4 BEGIN
100*C     4      WRITELN(FF,TD);
101*C     4      IF LBSMAX=0 THEN
102*C     5      REPLY:='M';
103*C     4      WRITE(' Enter Design Limit Load in pounds - ');
104*C     4      READLN(DLL);
105*C     4      DLL:=ABS(DLL);
106*C     4      TUL:=ROUND(1.5*DLL);
107*C     4      MAX_LD:=ROUND(LBSMAX/TUL*100);
108:C     4      END;
109:S
110*C     3      IF (REPLY='M') OR (REPLY='m') THEN

```

```

111:C      4      BEGIN
112:C      4      WRITELN(FF,TD);
113:C      4      WRITE(' Enter the maximum test load in pounds - ');
114:C      4      READLN(POUNDS);
115:C      4      LBSMAX:=ROUND(ABS(POUNDS));
116:C      4      MAX_LD:=ROUND(LBSMAX/TUL*100);
117:C      4      IF FS=0 THEN
118:C      5      REPLY:='V';
119:C      4      END;
120:S
121:C      3      IF (REPLY='V') OR (REPLY='v') THEN
122:C      4      BEGIN
123:C      4      WRITELN(FF,TD);
124:C      4      WRITE(' What is corresponding ',
125:C      4      ' voltage for 500kips? ');
126:C      4      READLN(FS);
127:C      4      FS:=ABS(FS);
128:C      4      END;
129:C      3
130:C      3      UNTIL REPLY=' ';
131:C      2
132:C      2 END; ( Procedure Chng_Syst_Param )
133:S
134:S
135:S
136:D      -56 1      ( ***** CHNG_TEST_PARAM ***** )
137:S
138:D      1      PROCEDURE CHNG_TEST_PARAM (VAR LDRATE,LD_TOL:INTEGER);
139:S
140:D      2      LABEL 10;
141:S
142:D      2      VAR
143:D      -4 2      MAX_LD :INTEGER;
144:D      -12 2      POUNDS :REAL;
145:D      -13 2      REPLY :CHAR;
146:S
147:C      2      BEGIN
148:C      2      IF LDRATE=0 THEN
149:C      3      BEGIN
150:C      3      REPLY:='L';
151:C      3      GOTO 10;
152:C      3      END;
153:S
154:C      2      REPEAT
155:S
156:C      3      WRITELN(FF,TD,TB,'Loading Rate is ',
157:C      3      LDRATE:6,' lbs/sec',LF);
158:C      3      WRITELN(TB,' Tolerance is ',LD_TOL:6,' lbs',TD);
159:S
160:C      3      WRITELN(' Hit:');
161:C      3      WRITELN(' L to change Loading rate');
162:C      3      WRITELN(' T to change Tolerance');
163:C      3      WRITE (' <space> to exit ');
164:C      3      REPEAT
165:C      4      READ(REPLY);

```

```

166*C      4      WRITE(BS,SP,BS);
167*C      4      UNTIL (((REPLY='L') OR (REPLY='1')) OR
168:C      4          ((REPLY='T') OR (REPLY='t')) OR
169:C      4          (REPLY=' '));
170:S
171*C      3      10:IF (REPLY='L') OR (REPLY='1') THEN
172:C      4          BEGIN
173*C      4              WRITELN(FF,TD);
174*C      4              WRITE ('      What is the ',
175:C      4                  'loading rate in lbs/sec? ');
176*C      4              READLN (LDRATE);
177*C      4              LDRATE:=ABS(LDRATE);
178*C      4              IF LD_TOL=0 THEN
179*C      5                  REPLY:='T';
180:C      4          END;
181:S
182*C      3      IF (REPLY='T') OR (REPLY='t') THEN
183:C      4          BEGIN
184*C      4              WRITELN(FF,TD);
185*C      4              WRITE(' Enter tolerance in pounds - ');
186*C      4              READLN(LD_TOL);
187*C      4              LD_TOL:=ABS(LD_TOL);
188:C      4          END;
189:S
190*C      3      UNTIL REPLY=' ';
191:C      2
192*C      2  END; { Procedure Chng_Test_Param }
193:S
194:S
195:D      -56  1  { ***** NUMBER_PASSES ***** }
196:S
197:D      1  PROCEDURE NUMBER_PASSES (VAR TOTPASS:INTEGER);
198:C      2  BEGIN
199:C      2      WRITELN(FF,TD);
200*C      2      WRITE ('      How many passes to be run? ');
201*C      2      READLN (TOTPASS);
202*C      2  END; { Proceedure Number_Passes }
203:S
204:S
205:D      -56  1  { ***** RESTART ***** }
206:S
207:D      1  PROCEDURE RESTART (VAR PASS,RECRD,POINT,TOTPASS: INTEGER);
208:D      2
209:D      2  LABEL 1,2;
210:S
211:D      2  VAR
212:D      -1  2      REPLY:CHAR;
213:S
214:C      2  BEGIN
215*C      2      IF TOTPASS<=0 THEN
216*C      3          NUMBER_PASSES (TOTPASS);
217*C      2      IF (RECRD=48) AND (POINT=200) THEN
218:C      3          BEGIN
219*C      3              PASS:=PASS+1;
220*C      3              RECRD:=1;

```

```

221*C      3      POINT:=1;
222*C      3      GOTO 2;
223:C      3      END;
224:S
225*C      2      IF PASS>0 THEN
226*C      3      GOTO 2;
227:S
228*C      2 1: WRITELN(FF,LF,LF);
229:C      2
230:C      2      { Find Pass number }
231*C      2      IF TOTPASS=1 THEN
232*C      3      PASS:=1
233:C      3      ELSE
234:C      3      REPEAT
235*C      4      WRITELN(TD);
236*C      4      WRITELN(TB,'Enter the Pass number ');
237*C      4      WRITE (TB,TB,'between 1 and ',TOTPASS:3,' - ');
238*C      4      READ (PASS);
239*C      4      UNTIL ((PASS>0) AND (PASS<=TOTPASS));
240:C      2
241:C      2      { Find Record number }
242:C      2      REPEAT
243*C      3      WRITELN(LF,TB,'Enter the Record number');
244*C      3      WRITE (LF,TB,TB,'between 1 and 48 - ');
245*C      3      READ (RECRD);
246*C      3      UNTIL ((RECRD>0) AND (RECRD<49));
247:S
248:C      2      { Find Point number }
249:C      2      REPEAT
250*C      3      WRITELN;
251*C      3      WRITELN(TB,'Enter the Point number ');
252*C      3      WRITE (LF,TB,TB,'between 1 and 200 - ');
253*C      3      READLN (POINT);
254*C      3      UNTIL ((POINT>0) AND (POINT<201));
255:S
256*C      2 2: WRITELN(FF,TD);
257*C      2      WRITELN(TB,'      Pass Number =',PASS:4,LF);
258*C      2      WRITELN(TB,'      Record Number =',RECRD:4,LF);
259*C      2      WRITELN(TB,'      Point Number =',POINT:4,LF);
260*C      2      WRITE (TB,'Restart at this point? (Y or N) - ');
261:C      2      REPEAT
262*C      3      READ (REPLY);
263*C      3      WRITE (BS,SP,BS);
264*C      3      IF (REPLY='N') OR (REPLY='n') THEN
265*C      4      GOTO 1;
266*C      3      UNTIL ((REPLY='Y') OR (REPLY='y'));
267:S
268*C      2      IF PASS>TOTPASS THEN
269*C      3      TOTPASS:=PASS;
270*C      2 END; { Procedure Restart }
271:S
272:S
273:D      -56 1      { ***** READ_CONFIG ***** }
274:S
275:D      1 PROCEDURE READ_CONFIG (FATIGUE:BOOLEAN;

```

```

276:D      2      VAR FS:REAL; VAR LBSMAX,DLL,LDRATE,LD_TOL,CAL_DELAY,
277:D      2      PASS,RECRD,POINT,TOTPASS:INTEGER);
278:S
279:D      2  VAR
280:D      -8  2      I,IOR      :INTEGER;
281:D      -9  2      REPLY      :CHAR;
282:D      -52 2      FILENAME,OLDNAME:S20;
283:D      -716 2      CONFIG_FILE :TEXT;
284:S
285:C      2  BEGIN
286:C      2      IF FATIGUE THEN
287:C      3          OLDNAME:='HSWRFCFG'
288:C      3      ELSE
289:C      3          OLDNAME:='HSWRSCFG';
290:C      2      DELETE(FILENAME,1,LENGTH(FILENAME));
291:C      2      WRITELN(FF,TD,TD);
292:C      2      WRITELN(T5,'What is file name to be read?',LF);
293:C      2      IF FATIGUE THEN
294:C      3          WRITE (T5,'Default is ''HSWRFCFG'' - ');
295:C      3      ELSE
296:C      3          WRITE (T5,'Default is ''HSWRSCFG'' - ');
297:C      2      READLN (FILENAME);
298:C      2      IF LENGTH(FILENAME)<1 THEN
299:C      3          FILENAME:=OLDNAME;
300:C      2      FILENAME:=CONCAT('#3:',FILENAME);
301:C      2      FILENAME:=CONCAT(FILENAME,'.TEXT');
302:C      2      $IOCHECK OFF$
303:C      2      RESET(CONFIG_FILE,FILENAME);
304:C      2      $IOCHECK ON $
305:C      2      IOR:=IORRESULT;
306:C      2      IF IOR<>0 THEN
307:C      3      BEGIN
308:C      3          WRITELN(TD,T5,'Can''t open file ',FILENAME,',',LF);
309:C      3          WRITELN(T5,'IORresult= ',IOR:1,LF);
310:C      3          WRITE (T5,'Hit any key to continue....');
311:C      3          READ(REPLY);
312:C      3      END
313:C      3      ELSE
314:C      3      BEGIN
315:C      3          READ(CONFIG_FILE,FS,LBSMAX,DLL,
316:C      3          LDRATE,LD_TOL,CAL_DELAY,
317:C      3          PASS,RECRD,POINT,TOTPASS);
318:C      3          CLOSE(CONFIG_FILE);
319:C      3      END;
320:S
321:C      2  END;  { Procedure Read_Config }
322:S
323:S
324:D      -56  1  { ***** WRITE_CONFIG ***** }
325:S
326:D      1  PROCEDURE WRITE_CONFIG (FATIGUE:BOOLEAN; FS:REAL;
327:D      2      LBSMAX,DLL,LDRATE,LD_TOL,CAL_DELAY,
328:D      -8  2      PASS,RECRD,POINT,TOTPASS:INTEGER);
329:S
330:D      -8  2  VAR

```



```

331:D   -12  2   IOR           :INTEGER;
332:D   -13  2   REPLY         :CHAR;
333:D   -56  2   FILENAME,OLDNAME:S20;
334:D  -720  2   CONFIG_FILE   :TEXT;
335:S
336:C           2 BEGIN
337*C           2   IF FATIGUE THEN
338*C           3     OLDNAME:='HSWRFCFG'
339:C           3   ELSE
340*C           3     OLDNAME:='HSWRSCFG';
341*C           2   DELETE(FILENAME,1,LENGTH(FILENAME));
342*C           2   WRITELN(FF,TD,TD);
343*C           2   WRITELN(T5,'What is file name to be written?',LF);
344*C           2   IF FATIGUE THEN
345*C           3     WRITE (T5,'Default is ''HSWRFCFG'' - ');
346:C           3   ELSE
347*C           3     WRITE (T5,'Default is ''HSWRSCFG'' - ');
348*C           2   READLN (FILENAME);
349*C           2   IF LENGTH(FILENAME)<1 THEN
350*C           3     FILENAME:=OLDNAME;
351*C           2   FILENAME:=CONCAT('#3:',FILENAME);
352*C           2   FILENAME:=CONCAT(FILENAME,'.TEXT');
353:C           2   $IOCHECK OFF$
354*C           2   REWRITE(CONFIG_FILE,FILENAME);
355:C           2   $IOCHECK ON $
356*C           2   IOR:=IORRESULT;
357*C           2   IF IOR<>0 THEN
358:C           3   BEGIN
359*C           3     WRITELN(TD,T5,'Can''t open file ',FILENAME,',',LF);
360*C           3     WRITELN(T5,'IORresult= ',IOR:1,LF);
361*C           3     WRITE (T5,'Hit any key to continue....');
362*C           3     READ(REPLY);
363:C           3   END
364:C           3   ELSE
365:C           3   BEGIN
366*C           3     WRITELN(CONFIG_FILE,FS,LBSMAX,DLL,
367:C           3       LDRATE,LD_TOL,CAL_DELAY,
368:C           3       PASS,RECRD,POINT,TOTPASS);
369*C           3     CLOSE (CONFIG_FILE,'SAVE');
370:C           3   END;
371:S
372*C           2 END; { Procedure Write_Config }
373:S
374:S
375:D  -56  1   { ***** PROGRAM_END ***** }
376:S
377:D           1 FUNCTION PROGRAM_END (FATIGUE:BOOLEAN):BOOLEAN;
378:S
379:D           2 VAR
380:D  -1    2   REPLY:CHAR;
381:S
382:C           2 BEGIN
383*C           2   WRITELN(BL,FF,TD);
384*C           2   WRITELN(' Are you sure you want to end the program?');
385*C           2   IF FATIGUE THEN

```

```

386*C      3      WRITE(' Configuration changes and/or the fatigue',
387:C      3      CR,LF,' test counters')
388:C      3      ELSE
389*C      3      WRITE(' Configuration changes');
390*C      2      WRITELN(' will be lost....',TD);
391*C      2      WRITE (' End the program(Y or N)? ');
392:C      2      REPEAT
393*C      3      READ (REPLY);
394*C      3      WRITE (BS,SP,BS);
395*C      3      UNTIL (REPLY='Y') OR (REPLY='y') OR
396:C      3      (REPLY='N') OR (REPLY='n');
397:S
398*C      2      PROGRAM_END:= ((REPLY='Y') OR (REPLY='y'));
399:S
400*C      2 END; { Function Program_End }
401:S
402:S
403:S
404:S
405:C      1 BEGIN
406:S
1000:C      1 $Linenum 1000$
1001:S
1002*C      1      FATIGUE:=TRUE;
1003*C      1      CONFIG_CHNG:=FALSE;
1004:C      1 10:REPEAT
1005:C      2      { Write menu }
1006*C      2      IF FATIGUE THEN
1007:C      3      BEGIN
1008*C      3      WRITELN(FF,FATIGUETITLE,LF,LF);
1009*C      3 {0} WRITELN(STRO,'Static Test',LF);
1010:C      3      END
1011:C      3      ELSE
1012:C      3      BEGIN
1013*C      3      WRITELN(FF,STATIC_TITLE,LF,LF);
1014*C      3      WRITELN(STRO,'Fatigue Test',LF);
1015:C      3      END;
1016*C      2 {1} WRITELN(STR1,LF);
1017*C      2 {2} WRITELN(STR2,LF);
1018*C      2 {3} WRITELN(STR3,LF);
1019*C      2 {4} WRITELN(STR4,LF);
1020:C      2 {5} (IF DUMPED THEN)
1021*C      2      WRITELN(STR5);
1022:C      2      WRITELN;
1023:S
1024*C      2 {6} WRITELN(STR6,LF);
1025*C      2 {7} WRITELN(STR7,LF);
1026*C      2 {8} IF FATIGUE THEN
1027:C      3      BEGIN
1028*C      3      WRITE (STR8,'Change number of passes');
1029*C      3      IF TOTPASS>0 THEN
1030*C      4      WRITE(' currently ',TOTPASS:1);
1031*C      3      WRITELN(LF);
1032:C      3      END
1033:C      3      ELSE

```

```

1034*C      3      WRITELN(STR8,LF);
1035:S
1036:C      2      {IF NOT(DUMPED) THEN}
1037*C      2 {9}    WRITELN(STR9);
1038:S
1039:C      2      { Read request }
1040*C      2      WRITE  (LF,'Request?');
1041*C      2      READ   (REPLY);
1042*C      2      WRITELN(FF);
1043:S
1044*C      2 {0}    IF REPLY='0' THEN
1045*C      3      FATIGUE:=NOT(FATIGUE);
1046:S
1047*C      2 {1}    IF REPLY='1' THEN
1048*C      3      READ_CONFIG (FATIGUE,FS,LBSMAX,DLL,LDRATE,
1049:C      3      LD_TOL,CAL_DELAY,PASS,RECRD,POINT,TOTPASS);
1050:S
1051*C      2 {2}    IF REPLY='2' THEN
1052:C      3      BEGIN
1053*C      3      CHNG_SYST_PARAM (FS,LBSMAX,DLL);
1054*C      3      CONFIG_CHNG:=TRUE;
1055:C      3      END;
1056:S
1057*C      2 {3}    IF REPLY='3' THEN
1058:C      3      BEGIN
1059*C      3      CHNG_TEST_PARAM (LDRATE,LD_TOL);
1060*C      3      CONFIG_CHNG:=TRUE;
1061:C      3      END;
1062:S
1063*C      2 {4}    IF REPLY='4' THEN
1064:C      3      BEGIN
1065*C      3      WRITE_CONFIG (FATIGUE,FS,LBSMAX,DLL,LDRATE,
1066:C      3      LD_TOL,CAL_DELAY,PASS,RECRD,POINT,TOTPASS);
1067*C      3      CONFIG_CHNG:=FALSE;
1068:C      3      END;
1069:S
1070*C      2 {5}    IF REPLY='5' THEN { Normally Power_Up }
1071:C      3      BEGIN
1072*C      3      WRITELN(FF,TD,T5,
1073:C      3      'This program is for checking the routines',LF);
1074*C      3      WRITELN(T5,'in TEST_PARM. ',
1075:C      3      'Use HSWRPC for checking the',LF);
1076*C      3      WRITELN(T5,'pump command routines.',TD);
1077*C      3      WRITE  (T5,'Hit any key to continue....');
1078*C      3      READ(REPLY);
1079*C      3      GOTO 10;
1080:C      3      END;
1081:S
1082*C      2 {6}    IF REPLY='6' THEN { Normally Cal load reading }
1083:C      3      BEGIN
1084*C      3      WRITELN(FF,TD,T5,
1085:C      3      'This program is for checking the routines',LF);
1086*C      3      WRITELN(T5,'in TEST_PARM. ',
1087:C      3      'Use HSWRLC for checking the',LF);
1088*C      3      WRITELN(T5,'load command routines.',TD);

```

Pascal [Rev 2.0 10/19/82] HSWRTP.TEXT

23-Nov-87 14:55:57 Page 10

```

1089*C      3      WRITE (T5,'Hit any key to continue....');
1090*C      3      READ(REPLY);
1091*C      3      GOTO 10;
1092:C      3      END;
1093:S
1094*C      2 {7}  IF REPLY='7' THEN
1095*C      3      IF CONFIG_CHNG THEN
1096:C      4      BEGIN
1097*C      4      IF PROGRAM_END(FATIGUE) THEN
1098*C      5      GOTO 90
1099:C      5      END
1100:C      4      ELSE
1101*C      4      GOTO 90;
1102:S
1103*C      2 {8}  IF (REPLY='8') AND FATIGUE THEN
1104:C      3      BEGIN
1105*C      3      NUMBER_PASSES(TOTPASS);
1106*C      3      CONFIG_CHNG:=TRUE;
1107:C      3      END;
1108:S
1109*C      2 {9}  UNTIL (REPLY='9') (AND NOT(DUMPED));
1110:C      1
1111*C      1      IF FATIGUE THEN
1112:C      2      BEGIN
1113*C      2      RESTART (PASS,RECRD,POINT,TOTPASS);
1114*C      2      CONFIG_CHNG:=TRUE;
1115:C      2      END;
1116:S
1117*C      1      GOTO 10;
1118:S
1119*C      1 90:WRITELN(CR,TD,TB,'Program Ended....');
1120:S
1121*C      1 END.
1122:S
1123:S

```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX C.4

CSC-2.4 Check Switches

@44 @44 @44 @44 @44 @44 @44 @44 @4X4 @44 @44 @44 @44 @44 @44 @44 @4X4Pascal
 [Rev 2.0 10/19/82] HSWRSC.TEXT 18-Nov-87 08:50:04 Page 1

```

1:D      0 $UCSD$ $LINES 55$ $DEBUG ON$
2:D      0 $SEARCH '#47:PASC_LIB.', '#3:VAR_DEFIN'$
3:S
4:D      0 PROGRAM HSWR_CHECK_SWITCHES;
5:D      1
6:D      1 { This program checks the six          }
7:D      1 { switches and the pump interlock      }
8:D      1 { connections on the counter panel    }
9:D      1 { Notes:                             }
10:D     1 { - Switch are numbered 0 to 5       }
11:D     1 { - Switch numbers correspond       }
12:D     1 { directly to bit numbers           }
13:D     1 { - Pin no. on card = SWITCH_NO + 1  }
14:D     1 { - Exp(Ln(2)*I)=2^I                }
15:D     1 { - Pump interlock is Switch #6     }
16:D     1 { - PASC_LIB and VAR_DEFIN required }
17:S
18:D     1 { Revision dated 12 November 1987   }
19:S
20:D     1 LABEL 10;
21:S
22:D     1 IMPORT
23:D     1   IODECLARATIONS, GENERAL_0,
24:D     1   GENERAL_1, VAR_DEFINITIONS;
25:S
26:D     1 CONST
27:D     1   ON ='  On  ';
28:D     1   OFF='  Off';
29:S
30:D     1 VAR
31:D    -8 1   I, SWITCH_NO: INTEGER;
32:D    -9 1   REPLY: CHAR;
33:S
34:D    -9 1 { MODULE PUMP_COMMANDS }
35:S
36:D    -9 1 { ***** DUMPED ***** }
37:S
38:D     1 FUNCTION DUMPED: BOOLEAN;
39:S
40:D     2 VAR
41:D    -4 2   WORD7: INTEGER; { 7 bit word }
42:S
43:C     2 BEGIN
44*C     2   IOCONTROL(12,3,-3792); { IEN,SYE,TME On }
45*C     2   IOCONTROL(12,1,1);
46*C     2   WRITEWORD(12,-3936); { ISL,SYE On }
47*C     2   IOCONTROL(12,3,0);
48*C     2   WORD7:=4095-(IOSTATUS(12,3)+32768);
49*C     2   DUMPED:=WORD7<64; { 64=2^6 }
50*C     2 END; { Procedure Dumped }
51:S
52:S
53:D    -9 1 { MODULE TEST_CONDITIONS }
54:S
55:D    -9 1 { ***** SWITCH_SET ***** }

```

```

56:S
57:D      1 FUNCTION SWITCH_SET (SWITCH_NO:INTEGER):BOOLEAN;
58:D      2   { This function returns value of True if the switch }
59:D      2   { number on Event Sense card SWITCH_NO is set
      }
60:D      2   { Notes:                                     }
61:D      2   {   - Pin no. on card = SWITCH_NO + 1       }
62:D      2   {   - Exp(Ln(2)*I)=2^I                       }
63:D      2   {   - Pump interlock is Switch #6           }
64:S
65:D      2 VAR
66:D      -8 2   I,WORD7:INTEGER;
67:D      -16 2   Ln2:REAL;
68:S
69:C      2 BEGIN
70*C      2   Ln2:=Ln(2);
71*C      2   IOCONTROL(12,3,-3792); { IEN,SYE,TME On }
72*C      2   IOCONTROL(12,1,1);
73*C      2   WRITEWORD(12,-3936);
74*C      2   IOCONTROL(12,3,0);
75*C      2   WORD7:=4095-(IOSTATUS(12,3)+32768); {4095=7777H}
76*C      2   FOR I:=6 DOWNT0 SWITCH_NO+1 DO
77*C      3     IF WORD7>=Exp(Ln2*I) THEN {2^I}
78*C      4     WORD7:=WORD7-ROUND(Exp(Ln2*I)); {2^I}
79*C      2   SWITCH_SET:=WORD7>=Exp(Ln2*SWITCH_NO); {2^SWITCH_NO}
80*C      2 END; { Function Set_Switch }
81:S
82:S
83:S
84:S
85:S
86:S
87:C      1 BEGIN
88*C      1   IOINITIALIZE;
89*C      1   WRITELN(FF,LF,LF,LF,LF,LF,LF,LF);
90*C      1   WRITELN(LF,LF,LF,LF,LF,LF,TB,'Hit any key to quit....');
91*C      1   WRITELN(US,US,US,US,US,US,US,US);
92*C      1   WRITELN(TB,TB,'Switch Number',LF,LF);
93*C      1   WRITELN(T5,' 0',1:5,2:5,3:5,4:5,5:5,T5,' Pump',LF);
94:C      1 10:REPEAT
95*C      2   WRITE (CR,SP,SP,SP);
96*C      2   FOR SWITCH_NO:=0 TO 6 DO
97*C      3     IF SWITCH_NO=6 THEN
98:C      4     BEGIN
99*C      4     WRITE(T5);
100*C     4     IF DUMPED THEN
101*C     5     WRITE (OFF)
102:C     5     ELSE
103*C     5     WRITE (ON);
104:C     4     END
105:C     4     ELSE
106*C     4     IF SWITCH_SET(SWITCH_NO) THEN
107*C     5     WRITE (ON)
108:C     5     ELSE
109*C     5     WRITE (OFF);
110:S

```

Pascal [Rev 2.0 10/19/82] HSWRSC.TEXT

18-Nov-87 08:50:04 Page 3

```
111*C      2    UNTIL NOT(UNITBUSY(KEYBRD));
112*C      1    READ(REPLY);
113*C      1    WRITE(BS,SP,BS);
114:S
115*C      1    Writeln(LF,LF,LF,LF,LF,LF);
116*C      1    Writeln(TB,'Program ended....');
117*C      1    IOUninitialize;
118*C      1    END.
119:S
120:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX C.5

CSC-2.5 Check Counters

```

1:D      0 $UCSD$ $LINES 55$ $DEBUG ON $
2:D      0 $SEARCH '#47:PASC_LIB.', '#3:VAR_DEFIN'$
3:S
4:D      0 PROGRAM HSWR_COUNTER_CHECK;
5:S
6:D      1 { This program checks the counters }
7:D      1 { and the pump interlock connections }
8:D      1 { on the counter panel. Note that }
9:D      1 { pump relay should always remain }
10:D     1 { closed during any communication }
11:D     1 { with the counters. }
12:S
13:D     1 { Revision dated 24 November 1987 }
14:S
15:D     1 LABEL 10;
16:S
17:D     1 IMPORT
18:D     1   IODECLARATIONS, GENERAL_0, GENERAL_1,
19:D     1   VAR_DEFINITIONS;
20:S
21:D     1 TYPE
22:D     1   COUNT_ARRAY=ARRAY[1..4] OF INTEGER;
23:S
24:D     1 VAR
25:D     -24 1   I, J, COUNTER, PASS, RECRD, POINT: INTEGER;
26:D     -40 1   COUNT: COUNT_ARRAY;
27:D     -41 1   REPLY: CHAR;
28:D     -42 1   DUMMY: BOOLEAN;
29:S
30:S
31:D     -42 1 { Note: Procedure Switch_Set is normally }
32:D     -42 1 { included in the TEST_COND Module, but is }
33:D     -42 1 { is not the purpose of this checkout rou- }
34:D     -42 1 { tine to check this procedure. Switch_Set }
35:D     -42 1 { is checked in the HSWRSC program, so }
36:D     -42 1 { don't copy this procedure to TEST_COND. }
37:D     -42 1 { The purpose of Switch_Set is to delay the }
38:D     -42 1 { CPU slightly, giving the Multiprogrammer }
39:D     -42 1 { time to open or close the relays on the }
40:D     -42 1 { Relay Output card. }
41:S
42:D     -42 1 { ***** SWITCH_SET ***** }
43:S
44:D     1 FUNCTION SWITCH_SET (SWITCH_NO: INTEGER): BOOLEAN;
45:D     2 { This function returns value of True if the switch }
46:D     2 { number on Event Sense card SWITCH_NO is set }
47:D     2 { Notes: }
48:D     2 { - Pin no. on card = SWITCH_NO + 1 }
49:D     2 { - Exp(Ln(2)*I)=2^I }
50:D     2 { - Pump interlock is Switch #6 }
51:S
52:D     2 VAR
53:D     -8 2   I, WORD7: INTEGER;
54:D     -16 2   Ln2: REAL;
55:S

```

```

56:C      2 BEGIN
57*C      2   Ln2:=Ln(2);
58*C      2   IOCONTROL(12,3,-3792); { IEN,SYE,TME On }
59*C      2   IOCONTROL(12,1,1);
60*C      2   WRITEWORD(12,-3936);
61*C      2   IOCONTROL(12,3,0);
62*C      2   WORD7:=4095-(IOSTATUS(12,3)+32768); {4095=7777H}
63*C      2   FOR I:=6 DOWNT0 SWITCH_NO+1 DO
64*C      3     IF WORD7>=Exp(Ln2*I) THEN {2^I}
65*C      4     WORD7:=WORD7-ROUND(Exp(Ln2*I)); {2^I}
66*C      2   SWITCH_SET:=WORD7>=Exp(Ln2*SWITCH_NO);{2^SWITCH_NO}
67*C      2 END; { Function Set_Switch }
68:S
69:S
70:S
71:D      -42  1 { ***** CLOSE_RELAY ***** }
72:S
73:D      1 PROCEDURE CLOSE_RELAY (COUNTER:INTEGER);
74:C      2 BEGIN
75:C      2   { Find bit corresponding to COUNTER }
76*C      2   IF COUNTER<0 THEN
77*C      3     COUNTER:=-COUNTER*2-1
78:C      3   ELSE
79*C      3     COUNTER:=(COUNTER-1)*2;
80*C      2   COUNTER:=ROUND(Exp(Ln(2)*COUNTER)); {2^I}
81:C      2   { Close the relay, but don't dump pump }
82:C      2   { Pump is 7th relay, 100 hex = 256 }
83*C      2   WRITEWORD(12,-4000);
84*C      2   WRITEWORD(12,Relay_Out_1+COUNTER+256);
85*C      2 END; { Proceedure Close_Relay }
86:S
87:S
88:D      -42  1 { ***** OPEN_RELAY ***** }
89:S
90:D      1 PROCEDURE OPEN_RELAY;
91:C      2 BEGIN
92:C      2   { Open the relay, but don't dump pump }
93*C      2   WRITEWORD(12,-4000);
94*C      2   WRITEWORD(12,Relay_Out_1+256);
95*C      2 END; { Proceedure Close_Relay }
96:S
97:S
98:D      -42  1 { ***** ZERO_COUNTERS ***** }
99:S
100:D     1 PROCEDURE ZERO_COUNTERS;
101:S
102:D     2 VAR
103:D     -1  2   DUMMY:BOOLEAN;
104:S
105:C     2 BEGIN
106:C     2   { Close the relays, but don't dump pump }
107*C     2   WRITEWORD(12,-4000);
108*C     2   WRITEWORD(12,Relay_Out_1+170+256); { 170=AA hex }
109*C     2   DUMMY:=SWITCH_SET(7);
110*C     2   OPEN_RELAY;

```

```

111*C      2 END;  { Procedure Zero_Counters }
112:S
113:S
114:D      -42  1  { ***** RESET_COUNTERS ***** }
115:S
116:D      1  PROCEDURE RESET_COUNTERS (PASS,RECRD,POINT:INTEGER);
117:S
118:D      2  VAR
119:D      -4  2    I:INTEGER;
120:D      -5  2    DUMMY:BOOLEAN; { Used as program delay as }
121:D      -5  2    { required by counters }
122:S
123:C      2  BEGIN
124:C      2    { Zero pass counter }
125*C      2    DUMMY:=SWITCH_SET(7);
126*C      2    CLOSE_RELAY (-3);
127*C      2    DUMMY:=SWITCH_SET(7); DUMMY:=SWITCH_SET(7);
128*C      2    OPEN_RELAY;
129:C      2    { Reset pass counter }
130*C      2    FOR I:=1 TO PASS DO
131:C      3    BEGIN
132*C      3      DUMMY:=SWITCH_SET(7);
133*C      3      CLOSE_RELAY (3);
134*C      3      DUMMY:=SWITCH_SET(7);
135*C      3      OPEN_RELAY;
136:C      3    END;
137*C      2    DUMMY:=SWITCH_SET(7);
138:S
139:C      2    { Reset record counter }
140*C      2    FOR I:=1 TO RECRD DO
141:C      3    BEGIN
142*C      3      DUMMY:=SWITCH_SET(7);
143*C      3      CLOSE_RELAY (2);
144*C      3      DUMMY:=SWITCH_SET(7);
145*C      3      OPEN_RELAY;
146:C      3    END;
147:S
148:C      2    { Reset point counter }
149*C      2    FOR I:=1 TO POINT DO
150:C      3    BEGIN
151*C      3      DUMMY:=SWITCH_SET(7);
152*C      3      CLOSE_RELAY (1);
153*C      3      DUMMY:=SWITCH_SET(7);
154*C      3      OPEN_RELAY;
155:C      3    END;
156*C      2 END;  { Procedure Reset_Counters }
157:S
158:S
159:S
160:D      -42  1  { ***** INCR_COUNTER ***** }
161:S
162:D      1  PROCEDURE INCR_COUNTER (COUNTER:INTEGER);
163:S
164:D      2  VAR
165:D      -1  2    DUMMY:BOOLEAN;

```

```

166:S
167:C      2 BEGIN
168*C      2     CLOSE_RELAY (COUNTER);
169*C      2     DUMMY:=SWITCH_SET(7); { Program delay }
170*C      2     OPEN_RELAY;
171*C      2 END; { Procedure Incr_Counter }
172:S
173:S
174:D      -42 1 { Note: Procedure Restart_Mod_HSWRCC is not }
175:D      -42 1 { in the TEST_COND Module, but is included }
176:D      -42 1 { in this checkout routine to check Proced- }
177:D      -42 1 { ure Reset_Counters. }
178:S
179:D      -42 1 { ***** RESTART_MOD_HSWRCC ***** }
180:S
181:D      1 PROCEDURE RESTART_MOD_HSWRCC (VAR PASS,RECRD,POINT:INTEGER);
182:D      2
183:D      2 LABEL 1,2;
184:S
185:D      2 VAR
186:D      -1 2     REPLY:CHAR;
187:S
188:C      2 BEGIN
189*C      2     IF PASS>0 THEN
190*C      3         GOTO 2;
191:S
192*C      2 1: WRITELN(FF,LF,LF);
193:C      2
194:C      2     { Find Pass number }
195*C      2     WRITELN(TD);
196*C      2     WRITE (TB,'Enter the Pass number - ');
197*C      2     READ (PASS);
198:C      2
199:C      2     { Find Record number }
200*C      2     WRITE (LF,TB,'Enter the Record number - ');
201*C      2     READ (RECRD);
202:S
203:C      2     { Find Point number }
204*C      2     WRITE (LF,TB,'Enter the Point number - ');
205*C      2     READLN (POINT);
206:S
207*C      2 2: WRITELN(FF,TD);
208*C      2     WRITELN(TB,' Pass Number =',PASS:4,LF);
209*C      2     WRITELN(TB,' Record Number =',RECRD:4,LF);
210*C      2     WRITELN(TB,' Point Number =',POINT:4,LF);
211*C      2     WRITE (TB,'Restart at this point? (Y or N) - ');
212:C      2     REPEAT
213*C      3         READ (REPLY);
214*C      3         WRITE (BS,SP,BS);
215*C      3         IF (REPLY='N') OR (REPLY='n') THEN
216*C      4             GOTO 1;
217*C      3     UNTIL ((REPLY='Y') OR (REPLY='y'));
218:S
219*C      2 END; { Procedure Restart_Mod_HSWRCC }
220:S

```

```

221:S
222:S
223:D      -42  1  { ***** Main Program ***** }
224:S
225:C      1 BEGIN
226*C      1   FOR I:=4 DOWNTD 1 DO
227:C      2   BEGIN
228*C      2     COUNT[I]:=0;
229*C      2     CLOSE_RELAY (-I);
230*C      2     DUMMY:=SWITCH_SET(7); DUMMY:=SWITCH_SET(7);
231:C      2   END;
232*C      1   OPEN_RELAY;
233:S
234:C      1   REPEAT
235*C      2     WRITELN(FF,LF,LF,LF);
236*C      2     WRITELN(T5,'Enter counter number you want changed,',LF);
237*C      2     WRITELN(T5,' '5' to change 1, 2, and 3 all at the same',l
F);
238*C      2     WRITELN(T5,'time (like restarting a fatigue test)',LF);
239*C      2     WRITELN(T5,'or 'Q' to quit....',LF,LF,LF);
240*C      2     WRITELN(TB,T5,'Counter 1 = ',COUNT[1]:1);
241*C      2     WRITELN(TB,TB,' 2 = ',COUNT[2]:1);
242*C      2     WRITELN(TB,TB,' 3 = ',COUNT[3]:1);
243*C      2     WRITELN(TB,TB,' 4 = ',COUNT[4]:1,LF);
244:S
245*C      2 10: READ(REPLY);
246*C      2   WRITE (BS,SP);
247:S
248*C      2   IF (REPLY>'0') AND (REPLY<'5') THEN
249:C      3   BEGIN
250*C      3     CASE REPLY OF
251*C      4       '1':COUNTER:=1;
252*C      4       '2':COUNTER:=2;
253*C      4       '3':COUNTER:=3;
254*C      4       '4':COUNTER:=4
255:C      4     END; { Case }
256*C      3     WRITELN(FF,TD,LF,TB,'How many times do you',LF);
257*C      3     WRITE (TB,'want counter ',REPLY,' incremented? ');
258*C      3     READLN(J);
259*C      3     IF J=0 THEN
260:C      4     BEGIN
261*C      4       COUNT[COUNTER]:=0;
262*C      4       CLOSE_RELAY (-COUNTER);
263*C      4       DUMMY:=SWITCH_SET(7); DUMMY:=SWITCH_SET(7);
264*C      4       OPEN_RELAY;
265:C      4     END
266:C      4   ELSE
267:C      4   BEGIN
268*C      4     COUNT[COUNTER]:=COUNT[COUNTER]+J;
269*C      4     FOR I:=1 TO J DO
270:C      5     BEGIN
271*C      5       INCR_COUNTER (COUNTER);
272*C      5       DUMMY:=SWITCH_SET(7);
273:C      5     END;
274:C      4   END;
275*C      3   IF (COUNTER=2) OR (COUNTER=3) THEN

```

Pascal [Rev 2.0 10/19/82] HSWRCC.TEXT

16-Dec-87 14:40:32 Page 6

```
276*C      4      COUNT[COUNTER-1]:=0;
277:C      3      END;
278:S
279*C      2      IF REPLY='5' THEN
280:C      3      BEGIN
281*C      3          RESTART_MOD_HSWRCC (PASS,RECRD,POINT);
282*C      3          RESET_COUNTERS (PASS,RECRD,POINT);
283*C      3          COUNT[1]:=POINT;
284*C      3          COUNT[2]:=RECRD;
285*C      3          COUNT[3]:=PASS;
286:C      3      END;
287:S
288*C      2      UNTIL (REPLY='Q') OR (REPLY='q');
289*C      1      WRITELN(LF,LF,LF);
290*C      1      WRITELN(TB,'Program ended....');
291:S
292*C      1      END.
293:S
```

No errors. No warnings.

***** Nonstandard language features enabled *****

APPENDIX C.6

CSC-2.6 Check Display


```

1:D      0 $UCSD$ $DEBUG ON $ $LINES 55$
2:D      0 $SEARCH '#47:PASC_LIB.', '#3:VAR_DEFIN'$
3:D      0
4:D      0 PROGRAM HSWR_CHECK_DISPLAY;
5:S
6:D      1 { This program is used to develop the crt }
7:D      1 { displays the operator will see during the }
8:D      1 { test. Note that some statements required }
9:D      1 { in the HSWR Program are left in this rou- }
10:D     1 { tine to make it easier to revise the main }
11:D     1 { program, but are "commented out" to simp- }
12:D     1 { lify this checking routine. All lines }
13:D     1 { "commented out" can be found by searching }
14:D     1 { for the string "+++". }
15:S
16:D     1 { Revision dated 22 December 1987 }
17:S
18:S
19:D     1 LABEL 25,30,40,45;
20:S
21:D     1 IMPORT
22:D     1 GENERAL_1, IODECLARATIONS,
23:D     1 SYSGLOBALS, CLOCK, VAR_DEFINITIONS;
24:S
25:D     1 TYPE
26:D     1 SPEC_ARRAY=ARRAY[1..10] OF INTEGER; {+++}
27:S
28:D     1 CONST
29:D     1 DLL=172090;
30:D     1 LBSMAX=300000;
31:D     1 FS=10.0;
32:D     1 CORR_VAL=1.08; { Correction to positive loads }
33:D     1 SPEC_VAL=SPEC_ARRAY[0,900,100,800,200,
34:D     1 700,300,600,400,500]; {+++}
35:S
36:D     1 VAR
37:D     1 I,J,IOR,
38:D     1 PASS,FLIGHT,POINT,
39:D     -32 1 FDBACK,LOADPT:INTEGER; { Loadpt=Nextpt in lbs }
40:D     -32 1 NEXTPC, { Corrected load point in %DLL }
41:D     -56 1 STRTPC,PC_DIF:REAL; { Previous load pt in %DLL }
42:D     -57 1 REPLY :CHAR;
43:D     -74 1 TO,T :REAL;
44:D     -78 1 TIME :TIMEREC;
45:D     -160 1 TITLE :S80;
46:D     -162 1 FATIGUE,REPLY_OK:BOOLEAN;
47:S
48:S
49:D     -162 1 { ***** LOAD ***** }
50:S
51:D     -8 1 FUNCTION LOAD (FS:REAL) :INTEGER;
52:S
53:D     -8 2 VAR
54:D     -12 2 V:INTEGER;
55:S

```

```

56:C      2 BEGIN
57:C      2   { Read voltage }
58*C      2   WRITEWORD(12,-3920); { ISL, SYE, TME On }
59*C      2   WRITEWORD(12,A_to_D_1);
60*C      2   READWORD (12,V);
61:C      2   { Convert volts to load }
62*C      2   V:=V+32768; { Subtract IRQ (100000oct) }
63*C      2   IF V>2047 THEN V:=V-4096; { 2's complement }
64:C      2   { LSB=5mV for 10V range, }
65:C      2   { FS=300,000lbs, 50/FS = resolution in pounds }
66*C      2   LOAD:=ROUND((0.005*V)*(300000/FS));
67:S
68*C      2 END;   { Function Load }
69:S
70:S
71:S
72:C      1 BEGIN
73*C      1   NEXTPC:=0;
74*C      1   LOADPT:=0;
75*C      1   STRTPC:=0;
76*C      1   PASS:=1;
77*C      1   FLIGHT:=1;
78*C      1   POINT:=1;
79*C      1   I:=1;
80*C      1   WRITE (FF,TD,T5,'Static or Fatigue (S or F)? ');
81:C      1   REPEAT
82*C      2     READ (REPLY);
83*C      2     WRITE(BS,SP,BS);
84*C      2   UNTIL (REPLY='S') OR (REPLY='s') OR
85:C      2     (REPLY='F') OR (REPLY='f');
86:S
87*C      1   FATIGUE:=((REPLY='F') OR (REPLY='f'));
88:S
89*C      1   TITLE:=STATIC_TITLE;
90*C      1   IF NOT(FATIGUE) THEN
91*C      2     GOTO 40;
92:C      1
93*C      1   TITLE:=FATIGUETITLE;
94*C      1   NEXTPC:=SPEC_VAL[I];
95:S
96:C      1   { ***** Compute Load Factors ***** }
97:S
98*C      1 25:LOADPT:=ROUND(NEXTPC*DLL/100);
99*C      1   WRITELN(FF,TITLE,LF,LF);
100:S
101:S
102*C      1   IF FATIGUE THEN
103:C      2   BEGIN
104*C      2     WRITELN(TB,'   Pass Number =',PASS:8,LF);
105*C      2     WRITELN(TB,'   Flight Number =',FLIGHT:8,LF);
106*C      2     WRITELN(TB,'   Point Number =',POINT:8,LF);
107*C      2     WRITELN(TB,'   Spectrum Value =',
108:C      2       (SPEC_VAL[I]/10):8:1,' %DLL',LF);
109*C      2     WRITELN(TB,'   Corrected Value =',NEXTPC:8:1,' %DLL',LF);
110:C      2   END

```

```

111:C      2      ELSE
112:C      2      BEGIN
113*C      2          WRITELN(TB,' Ramping to ',NEXTTPC:5:1,'% of DLL',LF);
114*C      2          WRITELN(TB,TB,' ',(2*NEXTTPC/3):5:1,'% of TUL',LF,LF);
115:C      2      END;
116:S
117*C      1      WRITELN(TB,' Desired Load =',LOADPT:8,' lbs',TD);
118*C      1      WRITELN(TB,' Hit any key to halt test....');
119*C      1      WRITE (TU,TB,' Feedback =',LOAD(FS):8,
120:C      1          ' lbs',BS,BS,BS,BS);
121:S
122:C      1      {+++ The following section is for checking routine }
123:S
124*C      1      30:SYSTIME(TIME);
125*C      1      WITH TIME DO
126*C      2          TO:=HOURL*3600+MINUTE*60+CENTISECOND/100;
127*C      1      T:=0;
128:S
129:C      1      REPEAT
130*C      2          WRITE (BT,LOAD(FS):8);
131*C      2          IF FATIGUE THEN
132:C      3              BEGIN
133*C      3                  SYSTIME(TIME);
134*C      3                  WITH TIME DO
135*C      4                      T:=HOURL*3600+MINUTE*60+CENTISECOND/100-TO;
136:C      3                  END;
137*C      2          UNTIL (NOT(UNITBUSY(KEYBRD)) OR (T>1));
138:S
139*C      1      IF T>1 THEN
140:C      2          BEGIN
141*C      2              STRTPC:=NEXTTPC;
142*C      2              I:=I+1;
143*C      2              IF I>10 THEN I:=1;
144*C      2              NEXTPC:=SPEC_VAL[I]*CORR_VAL/10;
145:C      2
146:C      2          { Increment Counters }
147*C      2          POINT:=POINT+1;
148*C      2          WRITE (TU,TU);
149*C      2          IF POINT=109 THEN { New Flight }
150:C      3              BEGIN
151*C      3                  POINT:=1;
152*C      3                  FLIGHT:=FLIGHT+1;
153*C      3                  WRITE (US,US);
154*C      3                  IF FLIGHT=356 THEN { New Pass }
155:C      4                      BEGIN
156*C      4                          FLIGHT:=1;
157*C      4                          PASS:=PASS+1;
158:C      4      {+++} {IF SWITCH_SET(1) THEN GOTO 40;} { Hold }
159*C      4                          WRITE (BT,US,US,PASS:8,LF,LF);
160:C      4                      END;
161:C      3      {+++} {IF SWITCH_SET(2) THEN GOTO 40;} { Hold }
162*C      3                          WRITE (BT,FLIGHT:8,LF,LF);
163:C      3                      END;
164:S
165*C      2          WRITE (BT,POINT:8,LF,LF);

```

```

166*C      2      WRITE (BT,(SPEC_VAL[1]/10):8:1,LF,LF);
167*C      2      WRITE (BT,NEXTPC:8:1,LF,LF);
168*C      2      LOADPT:=ROUND(NEXTPC*DLL/100);
169*C      2      WRITE (BT,LOADPT:8,LF,LF);
170:C      2
171*C      2      GOTO 30; { Go to Ramp }
172:C      2      END;
173:S
174*C      1      READ(REPLY);
175:S
176*C      1 40: WRITELN(FF,TITLE,LF,LF);
177*C      1      WRITELN(TB,' Holding at ',NEXTPC:5:1,'% of DLL',LF);
178*C      1      WRITELN(TB,TB,' ',(2*NEXTPC/3):5:1,'% of TUL',LF,LF);
179*C      1      WRITELN(TB,' Desired Load =',LOADPT:8,
180:C      1          ' lbs',TD);
181*C      1      IF NEXTPC<>0 THEN WRITELN(LF,LF,LF);
182*C      1      WRITELN(TB,' Hit:');
183:S
184*C      1      IF FATIGUE THEN
185*C      2          WRITELN(TB,' C to Continue test were stopped')
186:C      2      ELSE
187:C      2      BEGIN
188*C      2          WRITELN(TB,' S to trigger data Scan');
189*C      2          WRITELN(TB,' L for next Load point');
190:C      2      END;
191:S
192*C      1      IF NEXTPC=0 THEN
193*C      2          WRITELN(TB,' M to return to the Menu ',US)
194:C      2      ELSE
195*C      2          WRITELN(TB,' U to Unload ',TB,TU,US);
196:S
197*C      1      WRITE (TU,US);
198*C      1      IF NOT(FATIGUE) THEN WRITE(US);
199:S
200*C      1      FDBACK:=LOAD(FS);
201*C      1      WRITE (TB,' Feedback =',FDBACK:8,
202:C      1          ' lbs',BS,BS,BS,BS);
203*C      1      IF NEXTPC<>0 THEN
204:C      2      BEGIN
205*C      2          WRITELN(LF);
206*C      2 {+++} FDBACK:=LOADPT; { This statement for debugging }
207*C      2          WRITELN(T5,'Absolute Difference =',
208:C      2              (FDBACK-LOADPT):8,' lbs',LF);
209*C      2          PC_DIF:=ABS(1-FDBACK/LOADPT)*100;
210*C      2          IF PC_DIF<0.1 THEN PC_DIF:=0;
211*C      2          WRITE (T5,' Percent Difference =',
212:C      2              PC_DIF:8:1,' %',BS,BS,TU);
213:C      2      END;
214:S
215:C      1      { Inner loop repeats until operator }
216:C      1      { responds, and Outer loop repeats }
217:C      1      { until operator response is ok }
218:C      1 45: REPEAT { Outer loop }
219:C      2          REPEAT { Inner Loop }
220*C      3              FDBACK:=LOAD(FS);

```

```

221*C      3      WRITE (BT,FDBACK:B);
222*C      3      IF NEXTPC<>0 THEN
223:C      4      BEGIN
224*C      4          WRITE (LF,LF,BT,(FDBACK-LOADPT):B,LF,LF);
225*C      4          PC_DIF:=ABS(1-FDBACK/LOADPT)*100;
226*C      4          IF PC_DIF<0.1 THEN PC_DIF:=0;
227*C      4          WRITE (BT,PC_DIF:B:1,TU);
228:C      4      END;
229:S
230*C      3      SYSTIME (TIME);
231*C      3      WITH TIME DO
232*C      4          TO:=HOUR*3600+MINUTE*60+CENTISECOND/100;
233:C      3      REPEAT
234*C      4          SYSTIME (TIME);
235*C      4          WITH TIME DO
236*C      5              T:=HOUR*3600+MINUTE*60+CENTISECOND/100;
237*C      4          UNTIL (T-T0)>0.075; { 75 msec delay }
238:S
239:C      3          { Check for overload }
240:C      3          { +++ Taken out of checking routine }
241:S
242*C      3      UNTIL NOT(UNITBUSY(KEYBRD)); { Inner loop }
243:S
244*C      2      READ(REPLY);
245*C      2      WRITE(BS,SP,BS);
246:S
247*C      2      IF NEXTPC=0 THEN
248*C      3          REPLY_OK:=((REPLY='M') OR (REPLY='m'))
249:C      3      ELSE
250*C      3          REPLY_OK:=((REPLY='U') OR (REPLY='u'));
251:S
252*C      2      IF FATIGUE THEN
253:C      3      BEGIN
254*C      3          IF (REPLY='C') OR (REPLY='c') THEN
255*C      4              REPLY_OK:=TRUE;
256:C      3      END
257:C      3      ELSE
258*C      3          IF (REPLY='S') OR (REPLY='s') OR
259:C      4              (REPLY='L') OR (REPLY='l') THEN
260*C      4              REPLY_OK:=TRUE;
261:S
262:C      2      { Outer loop }
263*C      2      UNTIL REPLY_OK;
264:S
265:S      {+++IF (REPLY='M') OR (REPLY='m') THEN
266:C      1          GOTO 10;} { Go to Menu }
267:S
268*C      1      IF (REPLY='S') OR (REPLY='s') THEN
269:C      2      BEGIN
270:C      2      {+++ SCAN_1000 (TRUE);}
271*C      2          GOTO 45; { Continue hold }
272:C      2      END;
273:S
274*C      1      WRITELN(LF,LF);
275*C      1      IF NEXTPC<>0 THEN WRITELN(LF,LF,LF);

```

```

276:S
277*C      1      IF (REPLY='C') OR (REPLY='c') THEN
278:C      2      BEGIN
279*C      2          GOTO 25; { Go to Compute Load Factors }
280:C      2      END;
281:S
282*C      1      IF (REPLY='U') OR (REPLY='u') THEN
283:C      2      BEGIN
284*C      2          STRTPC:=NEXTPC;
285*C      2          NEXTPC:=0;
286*C      2          GOTO 25; { Go to Compute Load Factors }
287:C      2      END;
288:S
289*C      1      IF (REPLY='L') OR (REPLY='l') THEN
290:C      2      BEGIN
291*C      2          WRITELN(TB,TB,TB,TB);
292*C      2          WRITELN(LF,TB,TB,TB,TB);
293*C      2          WRITELN(TB,TB,TB,TB);
294*C      2          WRITELN(TB,TB,TB,TU);
295*C      2          WRITE  (BL,CR,'What is new load point ',
296:C      2              'in percent of DLL? ');
297:S
298*C      2          STRTPC:=NEXTPC;
299:S
300:C      2      REPEAT
301:C      3          $IOCHECK OFF$
302*C      3          READLN(NEXTPC);
303:C      3          $IOCHECK ON $
304*C      3          IOR:=IORESULT;
305*C      3          IF IOR<>0 THEN
306*C      4              WRITE  (CR,LF,'Try again, new load point? ');
307:C      4          ELSE
308*C      4              IF ABS(DLL*NEXTPC/100)>LBSMAX THEN
309:C      5                  BEGIN
310*C      5                      WRITELN(CR,LF,T5,'Maximum load is +/-',
311:C      5                          LBSMAX:1,'lbs (',ROUND(LBSMAX/DLL*100):1,
312:C      5                          '% DLL)',LF);
313*C      5                      WRITE  (T5,'try again, new load point? ');
314*C      5                      IOR:=1;
315:C      5                  END;
316*C      3          UNTIL IOR=0;
317:S
318*C      2      IF NEXTPC=STRTPC THEN
319:C      3      BEGIN
320*C      3          WRITE  (BL);
321*C      3          GOTO 40; { Go to Hold }
322:C      3      END;
323:S
324*C      2      GOTO 25; { Go to Compute Load Factors }
325:C      2      END;
326:S
327*C      1      WRITELN(TD);
328*C      1      WRITELN(TB,'Program Ended....');
329*C      1      END.
330:S

```

Pascal [Rev 2.0 10/19/82] HSWRDS.TEXT

22-Dec-87 11:16:21 Page 7

331:5

No errors. No warnings.

***** Nonstandard language features enabled *****

DISTRIBUTION LIST

No. of Copies

NAVAIRWARCENACDIWAR.....1
 Tom Hess, Code 60C1

NAVAIRWARCENACDIWAR.....1
 Code 6064

NAVAIRWARCENACDIWAR.....1
 Code 8131

NAVAIRWARCENACDIWAR.....25
 Code 6043

U 9400056

NAWCADWAR
 NAWCADWAR-93085-60

Date Due

NAVAL GENERAL LIBRARIES
 Chief of Naval Education
 and Training

NAVEDTRA 5070/2 (Rev. 9-80) S/N 0115-LF-050-7022

DISTRIBUTION LIST

No. of Copies

Director.....1
 Naval Research Laboratory
 Attn: L. Gause
 4555 Overlook Avenue, S.W.
 Washington, D.C. 20375-5000

Commander.....1
 Naval Air Warfare Center Weapons Division
 Attn: K. Bailey, Code 338
 China Lake, CA 93355

Commander.....1
 Naval Aviation Depot
 Attn: J. Gresham, Code 35210
 MCAS, CherryPoint, NC 28533-5030

Commander.....1
 Naval Aviation Depot
 Attn: D. Perl, Code 343
 NAS, North Island, San Diego, CA 92135-5112

Commander.....1
 Naval Aviation Depot
 Attn: D. Knapp, Code 342
 NAS, Pensacola, FL 32508-5300

Commander.....1
 United States Naval Academy
 Attn: Mechanical Engineering Dept.
 Annapolis, MD 21402

Commander.....1
 U.S. Naval Postgraduate School
 Attn: Technical Library
 Monterey, CA 93943

Commanding Officer.....1
 Warner Robbins Air Logistics Command
 Attn: T. F. Christian, MMSRD
 Robbins Air Force Base, GA 30198

Commanding Officer.....1
 U.S. Army Aviation Applied Technology Directorate
 Attn: Drew Orline
 SAV/TY-ATS
 Fort Eustis, VA 21604-5577

RECEIVED
 100 JUN 81

100056